mémento

CLEFS POUR LE VIC

Daniel Jean David

Editions du PSI

CLEFS POUR LE VIC

COLLECTION LANGAGES

Dictionnaire du Basic — David A. Lien, traduit par Yves Leclerc
Langages de programmation — Stéphane Berche et Claude Lhermitte
Le langage ADA — Daniel-Jean David
Programmer en APL — Daniel-Jean David
Programmer en Assembleur — Alain Pinaud
Programmer en Basic — Michel Plouin
Le Basic et ses Fichiers — tomes 1 et 2 — Jacques Boisgontier
Programmer en Fortran — Daniel-Jean David
Programmer en L.S.E. — Stéphane Berche et Yves Noyelle
Programmer en Pascal — Daniel-Jean David et Jean-Luc Deschamps
Comment programmer — Jean-Claude Barbance
Comprendre les microprocesseurs — Roland Dubois

COLLECTION MATERIELS

Un ordinateur personnel signé IBM — Pierre Lotigie-Laurent
La découverte de l'Applesoft — tome 1 — Frédéric Lévy et Dominique Schraen
La découverte de l'Applesoft — tome 2 — Frédéric Lévy
La pratique de l'Apple II — volumes I et II — Nicole Bréaud-Pouliquen
La pratique de l'Apple II — volume III — Nicole Bréaud-Pouliquen et Daniel-Jean David
La découverte du FX-702 P — Jean-Pierre Richard
La découverte du Goupil — Jean-Yves Michel
Programmer HP-41 — Philippe Descamps et Jean-Jacques Dhénin
La pratique du MZ-80 A et K — Jean-Pierre Lhoir
La découverte du PC-1211 — Jean-Pierre Richard
La découverte du PC-1500 — Jean-Pierre Richard
La découverte du CBM — Daniel-Jean David
La pratique du CBM — volumes I et II — Daniel-Jean David
La pratique du TRS-80 — volumes I, II et III — Pierre Giraud et Alain Pinaud
Exercices sur TRS-80 — Frédéric Lévy
Les graphiques sur TRS-80 — Don Inman, traduit par Alain Pinaud
La découverte de la TI-57 — Xavier de la Tullaye
La découverte du TI-99/4A — Frédéric Lévy et Dominique Schraen
La découverte du VIC — Daniel-Jean David
La pratique du VIC — volume I — Daniel-Jean David
La pratique du ZX81 — tome 1 — Xavier Linant de Bellefonds
La pratique du ZX81 — tome 2 — Marcel Henrot
Le petit livre du ZX81 — Trévor Toms, traduit par Ghislaine Lapeyre

COLLECTION GUIDES PRATIQUES

L'APL sur TRS-80 — Claude Nowakowski
CP/M pas à pas — Alain Pinaud
LISP sur Apple II — Nicole Bréaud-Pouliquen
La réalisation des programmes — Michel Benelfoul
Méthodes de calcul numérique — Claude Nowakowski
Pascal sur TRS-80 — Claude Nowakowski
Visicalc sur Apple — Hervé Thiriez
Visicalc sur TRS80 — Hervé Thiriez

COLLECTION PROGRAMMES

Collèges, poquettes et maths — Jacques Deconchat
Etudes pour ZX81 — tomes 1 et 2 — Jean-François Séhan
Jeux, trucs et comptes pour PET/CBM — Michel Benelfoul
Jeux, trucs et comptes pour TRS-80 — Michel Benelfoul et Denis Chauvin
Mathématiques et statistiques — Hervé Haut
Modèles pratiques de décision — tomes 1 et 2 — Jean-Pierre Blanger
Récréations pour TI-57 — tomes 1 et 2 — Jacques Deconchat
Suites pour PC-1500 — Jean-François Séhan
Variations pour PC-1211 — Jean-François Séhan

COLLECTION MEMENTOS

Clefs pour l'Apple II — Nicole Bréaud-Pouliquen
Clefs pour le CBM — Daniel-Jean David
Clefs pour le Vic — Daniel-Jean David
CP/M mot par mot
Le Basic de A à Z — Jacques Boisgontier

COLLECTION UTILISATIONS DE L'ORDINATEUR

Le Basic et l'Ecole — tomes 1 et 2 — Jacques Gouet
La comptabilité sur Apple II — Gérard Lilio et Serge Lilio
Les finances familiales — Jean-Claude Barbance
Outils financiers et comptables pour l'entreprise — Bernard Sulnon

INITIATION

Visa pour l'informatique — Jean-Michel Jégo
Mon ordinateur — Jean-Claude Barbance
L'ordinateur individuel — Yves Leclerc (en coédition avec S.C.E. Canada)

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les «copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective» et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, «toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite» (alinéa 1er de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

ISBN : 2-86595-074-3

© Editions du P.S.I., Parc industriel nord, Bâtiment 9, 77200 Torcy marne la Vallée

1983

Imprimé en France

CLEFS POUR LE VIC

par

Daniel Jean David

Editions du P.S.I.
1983

Daniel-Jean David *enseigne l'informatique de gestion à l'université de Paris 1 Panthéon Sorbonne.*

Par ailleurs il enseigne l'utilisation des microprocesseurs à l'E.N.S.A.M. de Paris.

Les sujets de recherche vont de l'informatique graphique aux techniques d'interface des microprocesseurs et aux systèmes multiprocesseurs.

Spécialiste du microprocesseur 6502, il a donné, à Paris, de nombreux séminaires sur les microprocesseurs, le KIM, le SYM et le P.E.T./C.B.M..

Il est directeur de "La Commode", magazine consacré aux ordinateurs Commodore.

Du meme auteur :

Aux Editions du P.S.I.

- La découverte du VIC
- La pratique du VIC - tome 1

P R E S E N T A T I O N

Ce livre est destiné à se trouver en permanence à côté de votre VIC lorsque vous l'utilisez. Il renferme toutes les informations de référence que vous pouvez souhaiter retrouver rapidement : syntaxe des commandes, codes caractères, messages d'erreur, langage machine, connecteurs, adresses utiles.

Les informations sont données sans détails excessifs, le but principal de ce livre étant l'accès rapide à l'information : pour un exposé plus introductif et plus complet, vous pouvez vous reporter à la série "La découverte du VIC", "La pratique du VIC"...

On termine par un recueil de "trucs" de différents niveaux mais tous utiles, les "Comment..." disposés sans ordre particulier, mais parmi lesquels vous pourrez vous retrouver rapidement grâce à un index alphabétique.

Nous serons reconnaissant à nos lecteurs de toute suggestion d'informations complémentaires à incorporer aux futures éditions de ce livre.

S O M M A I R E

CHAPITRES	Pages
I - **COMMANDES**	10
Fonctions et mots-clés du Basic	10
Répertoire des instructions et des opérateurs Basic	13
Commandes périphériques	17
Combinaisons de couleurs de l'écran	22
Optention des notes	23
Commandes de la cartouche programmer's aid	24
Commandes de la cartouche Super expander	27
II - **CARACTERES**	31
Codes caractères du VIC (en décimal)	31
Codes caractères du VIC (en hexadécimal)	35
Caractères de commande d'affichage	39
Représentation des instructions et des données Basic	40
III - **MESSAGES D'ERREUR**	43
Valeurs de ST en relation avec les cassettes	43
Valeurs de ST en relation avec l'IEEE	43
Messages d'erreur	43
Messages concernant les fichiers	47
Messages d'erreur sur disquettes	49
IV - **LANGAGE MACHINE 6502**	53
Registres internes du 6502	53
Jeu d'instructions du 6502	54
Schéma des modes d'adressage	59
Tableau de désassemblage	61
Jeu d'instructions du 6502	62
V - **CONNECTEURS**	65
Connecteurs	65
Brochage des principaux circuits intégrés utilisés	69
Registres des boîtiers d'entrée-sortie	72
VI - **ADRESSES**	77
Adresses fondamentales	77
Adresses des routines-système	78
Adresses des routines IEEE	86
Adresses des routines, RS 232	88
Routines d'entrée/sortie	89
Routines arithmétiques	90
Pointeurs fondamentaux	93
Adresses des variables stratégiques du VIC	94
Vecteurs	99

Conversion CBM/VIC 101
Conversion VIC/CBM 103

VII - **COMMENT ?** 105
Les "Comment...?" 105
Index alphabétique des "Comment...?" 117

FONCTIONS ET MOTS-CLES DU BASIC

Fonctions arithmétiques

ABS Valeur absolue de l'argument entre parenthèses.

ATN Arc tangente - le résultat est en radians, compris entre $-\pi/2$ et $\pi/2$.

COS Cosinus - l'argument doit être en radians. Exemple : cos(x en degrés) = COS (π*X/180).

EXP Exponentielle e^X. L'argument doit être < 88, sinon il se produit un dépassement de capacité.

FRE Quelle que soit la valeur de l'argument, fournit le nombre d'octets restés libres en mémoire. Exemple : juste après la mise sous tension, ?FRE(Ø) fournit 3581 et non 3583 car ?FRE(Ø) occupe 2 octets (cas d'un 5K).

INT Partie entière, ou plutôt plus grand entier inférieur ou égal à l'argument : INT(Ø.5) vaut Ø ; INT(5) vaut 5, INT(-Ø.5) vaut -1, INT(-3) vaut -3.

LOG Logarithme naturel (népérien, ou en base e). Pour obtenir le logarithme de X en base Y, utiliser LOG(X)/LOG(Y). Exemple : logarithme décimal de X = LOG(X)/LOG(10).

PEEK Fournit le contenu (compris entre 0 et 255) de la case mémoire dont l'adresse est égale à son argument (qui doit être entier et compris entre 0 et 65535).

POS POS(Ø) fournit la prochaine position d'impression libre sur la ligne d'écran (position du curseur).

RND Fournit un nombre pseudo-aléatoire compris entre 0 et 1. Voir l'explication détaillée au chap.5 de "La découverte du VIC".

SGN Fonction "signe" : 1 si X>0, -1 si X<0 et 0 si X=0

SIN Sinus - l'argument est supposé en radians.

SPC Ne peut s'employer que dans une instruction PRINT. PRINT SPC(I) imprime I espaces. I doit être entier compris entre 0 et 255.

SQR Racine carrée. L'argument doit être supérieur ou égal à 0.

TAB Ne peut s'employer que dans une instruction PRINT. PRINT TAB(I); fait aller à la position d'impression N° I (0 est la position la plus à gauche d'une ligne, 21 la plus à droite). I doit être compris entre 0 et 255. Si I<position où l'on est déjà, il n'y a pas d'action, c'est-à-dire que la prochaine impression se fera là où on était positionné.

TAN Tangente - l'argument est supposé en radians.

USR Appel d'un programme utilisateur en langage machine.

Fonctions chaînes

LEN(X$) Longueur

LEFT$(X$,N) Extraction à gauche

RIGHT$(X$,N) Extraction à droite

MID$(X$,K) ou **MID$(X$,K,N)** Extraction au milieu

ASC(X$) Conversion de caractère en ASCII

CHR$(K) Conversion d'ASCII en caractère

STR$(A) Représentation d'un nombre

VAL(X$) Valeur représentée par une chaîne

Mots-clés réservés en Basic

Mot clé	*Abrévia-tion (1)*	*Code déc.hex.*	*Mot clé*	*Abrévia-tion (1)*	*Code déc.hex.*
ABS	Ab	182 $B6	**LOAD**	Lo	147 $93
AND	An	175 $AF	**LOG**	LOg	188 $BC
ASC	As	198 $C6	**MIDS**	Mi	202 $CA
ATN	At	193 $C1	**NEW**	NEw	162 $A2
CHR$	Ch	199 $C7	**NEXT**	Ne	130 $82
CLOSE	CLo	160 $A0	**NOT**	No	168 $A8
CLR	Cl	156 $9C	**ON**	On	145 $91
CMD	Cm	157 $9D	**OPEN**	Op	159 $9F
CONT	Co	154 $9A	**OR**	Or	176 $B0
COS	COs	190 $BE	**PEEK**	Pe	194 $C2
DATA	Da	131 $83	**POKE**	Po	151 $97
DEF	De	150 $96	**POS**	POs	185 $B9
DIM	Di	134 $86	**PRINT**	?	153 $99
END	En	128 $80	**PRINT#**	Pr	152 $98
EXP	Ex	189 $BD	**READ**	Re	135 $87
FN	Fn	165 $A5	**REM**	REm	143 $8F
FOR	Fo	129 $81	**RESTORE**	REs	140 $8C
FRE	Fr	184 $B8	**RETURN**	REt	142 $8E
GET	Ge	161 $A1	**RIGHT$**	Ri	201 $C9
GOSUB	GOs	141 $8D	**RND**	Rn	187 $BB
GOTO	Go	137 $89	**RUN**	Ru	138 $8A
IF	If	139 $8B	**SAVE**	Sa	148 $94
INPUT		133 $85	**SGN**	Sg	180 $B4
INPUT#	In	132 $84	**SIN**	Si	191 $BF
INT	INt	181 $B5	**SPC**	Sp	166 $A6
LEFT$	LEf	200 $C8	**SQR**	Sq	186 $BA
LEN		195 $C3	**STEP**	STe	169 $A9
LET	Le	136 $88	**STOP**	St	144 $90
LIST	Li	155 $9B	**STR$**	STr	196 $C4

(1) Les minuscules représentent les lettres avec SHIFT. Elles apparaissent ainsi lorsqu'on est en mode texte.

Mot clé	*Abréviation (1)*	*Code déc.*	*hex.*
SYS	Sy	158	$9E
TAB	Ta	163	$A3
TAN	TAn	192	$C0
THEN	Th	167	$A7
TO	To	164	$A4
USR	Us	183	$B7
VAL	Va	197	$C5
VERIFY	Ve	149	$95
WAIT	Wa	146	$92

Codes opérateurs

+	−	*	/	↑	>	=
170 $AA	171 $AB	172 $AC	173 $AD	174 $AE	177 $B1	178 $B2

<
179 $B3

Comme le Guide Michelin, nous attribuons des étoiles aux instructions et aux commandes : *** instruction fondamentale, ** instruction importante, * instruction intéressante. Pas d'* : instruction qu'il ne faut pas hésiter à utiliser si besoin s'en fait sentir.

D'autre part P veut dire : plutôt instruction en mode programmé, Ø veut dire : interdit en mode programmé, C : plutôt commande en mode direct, Ø : interdit en mode direct, "rien" veut dire : tantôt l'un, tantôt l'autre.

La définition est suivie d'exemples non commentés montrant les différentes formes que peut prendre l'instruction.

Catégorie	*Mot-clé*	Définition-*exemples*
	CLOSE	Fermeture d'un fichier *CLOSE 5*
C	CLR	Remise à zéro de toutes les variables
	CMD	Commande de l'interface IEEE488
CØ	CONT	Continuer dans le programme après interruption
P*	DATA	Définit une liste de constantes qui seront "lues" par une instruction READ. *1Ø DATA ABC,DEF,5,3.25*
P*	DEF FN	Définition d'une fonction utilisateur. *1Ø DEF FNF(X)=A*X+B*
P*	DIM	Dimensionnement d'un tableau (fixe les valeurs maxima des indices). *1Ø DIM A(15),B(3),CØ(2,5),D(7,6,2)* *2Ø DIM W(N)*
PØ*	END	Termine un programme
**	FOR	Introduit une boucle : toutes les instructions comprises entre FOR I = A TO B STEP C et le NEXT correspondant seront répétées pour toutes les valeurs de I allant de A à B, C par C : *1Ø FOR I=1 TO 1ØØØ* *5Ø FOR I=Ø TO 2ØØ STEP 5* *6Ø FOR I=N TO 3*N+4 STEP 5* *7Ø FOR I=5Ø TO 1 STEP -1* *8Ø FOR X=1.5 TO 2*X STEP .1*

Catégorie	*Mot-clé*	Définition-*exemples*
PC*	GET	Saisit un caractère au clavier *1Ø GET A$*
P**	GOSUB	Appel d'un sous-programme *1Ø GOSUB 1ØØØ*
P**	GOTO	Saut à une autre instruction *1Ø GOTO 5Ø*
P**	IF	Saut conditionnel, de la forme : IF condition THEN instruction. Si la condition n'est pas satisfaite on passe à la ligne suivante ; si la condition est satisfaite, on effectue l'instruction qui suit THEN s'élide en IF c THEN GOTO x IF c THEN x IF c GOTO x ou en *1Ø IF A>B THEN Y=K* *2Ø IF A=3 GOTO 1ØØØ* *3Ø IF A$<>"" THEN 5Ø*
PØ***	INPUT	Acquisition de données au clavier *1Ø INPUT A* *2Ø INPUT A,B,C$,D* *3Ø INPUT"ENTREZ UN NOMBRE";N*
C***	LIST	Liste du programme *LIST LIST 1Ø- LIST -1ØØ* *LIST 1Ø LIST 1Ø-1ØØ*
C**	LOAD	Chargement d'un programme sur cassette. *LOAD LOAD"PROG"*
CP*	NEW	Vide la mémoire-programme
**	NEXT	Fait passer à l'itération suivante dans un FOR. *NEXT I NEXT J,I NEXT*
P	ON	*ON I GOTO 1Ø,2Ø,3Ø* Si I vaut 1, on va en 1Ø, s'il vaut 2, on va en 2Ø, 3Ø s'il vaut 3. *ON I GOSUB 1ØØØ, 15ØØ, 2ØØØ, 5ØØØ* Si I vaut 1, on appelle le sous-programme en 1ØØØ, 2 en 15ØØ, 3 en 2ØØØ, 4 en 5ØØØ

Catégorie	*Mot-clé*	Définition-*exemples*
P	OPEN	Ouverture d'un fichier *OPEN 1,4 imprimante* *OPEN 5,1,Ø,"TATA"* *OPEN L,P*
*	POKE	POKE a,b écrit la donnée b à l'adresse a *POKE 37148,252 POKE K+1,Z-4*
***	PRINT	Imprime un résultat sur l'écran. *PRINT A 1Ø PRINT A;B,J* *2Ø PRINT 2*A+3,B$* *3Ø PRINT"LE RESULTAT EST";B;*
P*	READ	"Lecture" de données dans une instruction DATA associée. *1Ø READ A 2Ø READ A,B$,C*
P	REM	Introduit un commentaire.
P	RESTORE	Revient au début des DATA
P**	RETURN	Retour de sous-programme *1ØØ RETURN*
C***	RUN	Déclenche l'exécution d'un programme. *RUN RUN 5ØØ*
C**	SAVE	Sauvegarde d'un programme sur cassette. *SAVE SAVE"TATA"*
**	STEP	Introduit le pas d'incrémentation dans un FOR.
P¢	STOP	Arrête l'exécution d'un programme *1Ø STOP*
	SYS	Fait démarrer l'exécution d'un programme en langage machine à l'adresse indiquée *SYS(648Ø2)*
P**	THEN	Introduit l'instruction à effectuer quand un IF est satisfait.
**	TO	Introduit la valeur limite d'un FOR.

Catégorie	*Mot-clé*	Définition-*exemples*
CP**	VERIFY	Vérification d'un programme qui vient juste d'être enregistré sur cassette. *VERIFY VERIFY"TATA"*
	WAIT	*WAIT A,B,C* (C=Ø si absent) Suspend l'exécution du programme jusqu'à ce que (contenu de l'adresse A) et (B) OU exclusif (C) soit égal à 1. *WAIT 653,1* attend qu'on appuie sur la touche 'Shift'
***	=	Affectation de valeur à une variable. *1Ø A=1 C$="TEXTE" 2Ø N=N+1*

Opérateurs

Arithmétiques

+ addition de nombres ou concaténation de chaînes de caractères

- prendre l'opposé ou soustraction

* multiplication

/ division

Comparaisons

= égal	<>différent
< inférieur	> supérieur
<=inf. ou =	>=sup. ou =
=<inf. ou =	=>sup. ou =

Logiques effectués bit à bit

NOT non logique, agit sur 1 seul opérande

AND et logique
OR ou logique
} 2 opérandes

bit 1	bit 2	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

bit	NOT
0	1
1	0

Le nombre d'étoiles marque l'importance de la commande.

1- Cassettes

***	LOAD "NOM"	charge le programme NOM à partir de la cassette.
	LOAD "NOM",1,1	chargement sans translation.
***	SAVE "NOM"	sauve le programme présent en mémoire sous le nom NOM, sur la cassette (par défaut).
**	VERIFY	compare le premier programme trouvé sur la cassette avec le contenu de la mémoire.
***	OPEN 1,1,1,"NOM"	ouvre en écriture un fichier qui s'appellera NOM sur la cassette (n° logique 1)
***	OPEN 5,1,Ø,"NOM"	ouvre en lecture le fichier NOM de la cassette (n° logique 5).
***	PRINT#1,ACHR(13);	écrit la chaîne A$ et un retour-chariot sur le fichier logique 1.
***	INPUT#5,X	lit le nombre X sur le fichier logique 5.
***	GET#5, C$	lit le caractère C$ sur le fichier logique 5.

2- Imprimante VIC 1515

***	OPEN 4,4	ouvre le fichier logique 4 sur l'imprimante avec, par défaut, l'adresse secondaire 0. Les données seront imprimées normalement en mode semi-graphique.
**	OPEN 4,4,7	ouvre le fichier avec l'adresse secondaire 7 (la seule admise par la 1515, à part 0). Les données seront imprimées normalement en mode texte (minuscules).

***	PRINT#4,A$	envoie la chaîne A$ vers l'imprimante (qui la traitera en fonction de l'adresse secondaire avec laquelle le fichier 4 a été ouvert).
**	CMD 4	définit le périphérique sur lequel a été ouvert le fichier logique 4 comme sortie standard du système.
***	CLOSE 4	fermeture du fichier 4.

Caractères spéciaux

*	Rvs[C]	CHR$(18)	contraste inversé
	Off[C]	CHR$(146)	retour au contraste normal
***		CHR$(13)	retour-chariot, nouvelle ligne
**		CHR$(1Ø)	
***		CHR$(14)	caractères élargis
***		CHR$(15)	retour à la normale (après élargissement du caractère programmable).
**	h	CHR$(145)	passage au mode graphique
**	b	CHR$(17)	passage au mode texte
***		CHR$(8)	passage au mode caractère programmable
		CHR$(26)	répétition du caractère programmable
**		CHR$(16)	positionnement (en caractères)
**		CHR$(27)+CHR$(16)	positionnement (en colonnes élémentaires).

3- Disquette VIC 1540

Sauvegarde - chargement de programmes

***	LOAD"Ø:NOM",8	charge le programme NOM depuis la disquette.
***	SAVE "↑Ø:NOM",8	sauve le programme NOM sur la disquette (@ signifie remplacement).

**	VERIFY "NOM",8	vérifie le programme NOM.

Commandes de manipulation de fichiers

***	OPEN 8Ø,8,15	ouvre le canal de commande.
**	INPUT#8Ø,EN,EM$,EP,ES	lit les données qui précisent quelle erreur s'est produite.
***	PRINT#8Ø,"commande"	envoie une commande.
**	"NØ:NOM,ID"(1)	formate la disquette Ø sous le nom NOM et l'identificateur ID.
***	"IØ"	
**	"I"	initialise la disquette.
**	"VØ"	nettoie les fichiers pendants de la disquette
**	"CØ:ZOZO=Ø:ZAZA"	copie le fichier ZAZA de la disquette sous le nom ZOZO.
*	"RØ:NONO=NANA"	le fichier NANA change de nom : il s'appellera NONO.
***	"SØ:ZOZO"	supprime le fichier ZOZO de la disquette.
	"SØ:X*"	supprime tous les fichiers de la disquette dont le nom commence par X.

Fichiers séquentiels

***	OPEN 82,8,2,"Ø:ZOZO,S,W"	ouvre le canal 2 sous le n° logique 82, attribué au fichier séquentiel ZOZO de la disquette, en écriture.
***	OPEN 83,8,3,"ZOZO"	ouvre le canal 3 sous le n° logique 83 pour y chercher en lecture le fichier séquentiel ZOZO.
***	INPUT#83,A$	lit la chaîne A$ sur le fichier logique 83.
*	GET#83,X$	lit le caractère X$ sur le fichier logique 83.

(1) PRINT#8Ø, est sous-entendu.

*** PRINT#82,ACHR(13); écrit la chaîne A$ sur le fichier logique 82.

*** CLOSE 83 ferme le fichier logique 83, et le canal 3.

*** CLOSE 8Ø ferme le canal 15 ainsi que tous les autres.

Ajout

** OPEN 82,8,2,"Ø:ZOZO,A" ouvre le fichier séquentiel ZOZO en se positionnant à la fin. Cela permet d'ajouter des enregistrements.

Accès direct

*** OPEN 82,8,2,"#" ouvre le canal d'accès direct 2 sous le n° logique 82.

*** PRINT#82,ACHR(13); écrit la chaîne A$ dans le tampon du canal 2.

*** INPUT#82,A$ lit la chaîne A$ dans le tampon du canal 2.

* GET#82,X$ lit le caractère X$ dans le tampon du canal 2.

*** PRINT# 8Ø,"commande" envoie une commande d'accès direct.

Commandes

*** "B-R" 5;Ø;15;3 lit sur le canal 5 le secteur 3 de la piste 15 de la disquette

*** "B-W" 3;7;Ø envoie le tampon du canal 3 sur le secteur Ø de la piste 7 de la disquette.

** "B-P" 2;1Ø se positionne sur le 10° octet du tampon du canal 2.

** "B-A" 1;15;2 alloue le secteur 2 de la piste 15 de la disquette 1.

* "B-F" 1;15;2 libère ce même secteur.

"M-W" CHR$(Ø) CHR$(18) CHR$(2) CHR$(74) CHR$(95) écrit 2 octets à partir de l'adresse 1200 hexadécimal de la mémoire de la 1540 : les valeurs 4A et 60 en hexadécimal.

"M-R" CHR$(Ø) CHR$(18) suivi de
GET#8Ø,X
obtient en X le contenu de l'adresse 1200 hexadécimal.

"M-E" CHR$(Ø) CHR$(18) fait exécuter à partir de 1200.

"B-E" 4;Ø;11;2 fait lire dans le canal 4 le secteur 2 de la piste 11 disquette Ø et fait commencer l'exécution.

** "U-2:" 2;1;15;3 met à jour le secteur 3 de la piste 15 de la disquette 1 à travers le canal 2.

Fichiers relatifs

*** OPEN 82,8,2,"Ø:ZOZO,L"+CHR$(5Ø) ouvre le fichier relatif ZOZO avec une taille d'enregistrement de 50 octets.

*** PRINT#8Ø,"P"+CHR$(2)+CHR$(5)+CHR$(3)+CHR$(1Ø) se positionne dans le fichier relatif ouvert sur le canal 2, au 10° octet de l'enregistrement n° 3*256+5=773.

4- Commandes du programme CAT

Commodore livre avec les unités de disques un programme appelé CAT destiné à faciliter l'usage des commandes disques.

Si CAT est le premier programme de la disquette, il se charge par LOAD "*",8 suivi de RUN.

Ensuite : >"chaîne de commande" a le même effet que :
OPEN 8,8,15 : PRINT#8, "chaîne"

Ainsi, >IØ fait initialiser la disquette.
>SØ:FICHIER fait supprimer le fichier FICHIER de la disquette.

répertoire : D fait lister le répertoire de la disquette mais c'est mieux que LOAD "$Ø",8 car là, cela ne *perturbe pas le programme éventuellement en mémoire*.

lecture d'une erreur : la commande S fournit le moyen d'avoir les renseignements voulus, lorsqu'une erreur s'est produite.

sortie du programme : la commande Q fait sortir de CAT.

POKE 36879 ,X avec X donnê par le tableau :

Cadre / Ecran	Noir	Blanc	Rouge	Turquoise	Pourpre	Vert	Bleu	Jaune
Noir	8	9	10	11	12	13	14	15
Blanc	24	25	26	27	28	29	30	31
Rouge	40	41	42	43	44	45	46	47
Turquoise	56	57	58	59	60	61	62	63
Pourpre	72	73	74	75	76	77	78	79
Vert	88	89	90	91	92	93	94	95
Bleu	104	105	106	107	108	109	110	111
Jaune	120	121	122	123	124	125	126	127
Orange	136	137	138	139	140	141	142	143
Orange clair	152	153	154	155	156	157	158	159
Rose	168	169	170	171	172	173	174	175
Turquoise clair	184	185	186	187	188	189	190	191
Pourpre clair	200	201	202	203	204	205	206	207
Vert clair	216	217	218	219	220	221	222	223
Bleu clair	232	233	234	235	236	237	238	239
Jaune clair	248	249	250	251	252	253	254	255

POKE 36878, volume (de 1 à 15, 0 = silence)
POKE voix, N avec voix 1 = 36874 (grave) voix 3 = 36876 (aigu)
voix 2 = 36875 (médium) voix 4 = 36877 (bruit)

Valeurs des notes pour voix 3

Note	*Fréquence (Hz)*	*N*	*Note*	*Fréquence*	*N*	*Note*	*Fréquence*	*N*
DO 2	130,79	non def	RE #	310,88	199	FA #	739,64	231-232
DO #	138,50	130	MI	329,60	202-203	SOL	783,73	233
RE	146,78	137	FA	349,22	205-206	SOL #	829,97	234
RE #	155,44	143-144	FA #	369,82	208	LA	880,00	235
MI	164,80	150	SOL	391,86	211	LA #	931,93	236-237
FA	174,61	156	SOL #	414,97	213	SI	987,57	237-238
FA #	184,91	161	LA	440,00	215-216	DO 5	1046,37	238-239
SOL	195,93	166-167	LA #	465,96	218	DO #	1108,10	239-240
SOL #	207,48	171-172	SI	493,88	220	RE	1174,02	240
LA	220,00	176	DO 4	523,19	222	RE #	1243,28	241
LA #	232,98	181	DO #	554,05	224	MI	1318,42	242
SI	246,94	185	RE	587,01	225-226	FA	1396,88	242-243
DO 3	261,59	189	RE #	621,66	227	FA #	1479,29	243
DO #	277,02	192-193	MI	659,21	229	SOL	1567,46	244
RE	293,56	196	FA	698,44	230	SOL #	1659,94	244-245

COMMODORE

COMMANDES DE LA CARTOUCHE PROGRAMMER'S AID

Lancement par SYS 7*4096+9 (SYS 28681)

Fonctions préassignées des touches

Mode PROG	*Normal*	*SHIFT*	*CTRL*
f1/f2	(1) LIST	(5) MID$(	(9) EDIT C/R
f3/f4	(2) RUN C/R	(6) LEFT$(	(10) GOSUB
f5/f6	(3) GOTO	(7) RIGHT$(	(11) RETURN
f7/f8	(4) INPUT	(8) CHR$(	(12) STR$(

Mode EDIT	*Normal*	*SHIFT*	*CTRL*
f1/f2	(1) LIST	(5) AUTO	(9) PROG C/R
f3/f4	(2) RUN C/R	(6) DELETE	(10) RENUMBER
f5/f6	(3) FIND	(7) CHANGE	(11) MERGE
f7/f8	(4) TRACEC/R	(8) STEPC/R	(12) OFF C/R

() Numéro de la touche pour la commande KEY
C/R Le retour chariot fait partie de la fonction.

Touches CTRL supplémentaires

CTRL **A** Déroulement vers le bas d'un listing.

CTRL **E** Sortie du mode guillemets.

CTRL **L** Supprime tous les caractères après le curseur dans la ligne.

CTRL **N** Vide l'écran après le curseur.

CTRL **Q** Déroulement d'un listing vers le haut.

CTRL **U** Vide la ligne.

Commandes

* AUTO N,I numérotation automatique : la première ligne est N, l'incrément I (valeur par défaut 10). On s'arrête par Return. La valeur par défaut de N est 'dernière valeur où on s'était arrêté.'

COMMANDES DE LA CARTOUCHE PROGRAMMER'S AID

CHANGE X,Y,A-B remplace X par Y dans les lignes A à B
Ex. CHANGE GOSUB, GOTO

* DELETE A-B suppression de lignes avec la même syntaxe que LIST sauf que DELETE tout court est interdit.

* DUMP supprime la valeur de toutes les variables (non indicées).

** EDIT passe en mode EDIT vis-à-vis des touches préassignées.

FIND X,A-B trouve la chaîne X dans le programme A et B. Ex. FIND PRINT.

* HELP localise précisément une erreur.

*** KEY n,"texte" assigne la chaîne "texte" à la touche de fonction n° n (voir tableau). A chaque fois qu'on appuiera sur la touche, la chaîne apparaîtra.
Ex. KEY 12,"BONJOUR"
KEY 1 ,"LIST"+CHR$(13)

KILL sort de Programmer's Aid.

*** MERGE "nom fichier",PERIPH charge le programme indiqué depuis le périphérique spécifié mais sans effacer le programme déjà en mémoire. Il y a fusion des deux programmes.
Ex. MERGE "Ø:ZOZO",8

* OFF sortie du mode TRACE ou du mode STEP.

** PROG passe en mode PROG vis-à-vis des touches préassignées.

* RENUMBER N,I renumérote tout le programme, la première ligne prenant le n° N, l'incrément étant I. Valeurs par défaut 10.

** STEP passe en mode STEP. Lorsqu'on fait RUN dans ce mode, les numéros de lignes exécutées apparaissent au coin supérieur droit de l'écran. Une ligne est exécutée à chaque fois qu'on appuie sur SHIFT.

** TRACE	passe en mode TRACE. Lorsqu'on fait RUN dans ce mode, les numéros de lignes exécutées apparaissent au coin supérieur droit de l'écran ; l'exécution est continue.

Le lancement est automatique. La cartouche ne peut être montée en même temps qu'un jeu ou qu'une extension RAM 3 K.

Commandes de lecture

KEY — lit les assignations des touches de fonction. (mode direct seulement).

MG=RGR(Ø) — donne pour valeur à MG le mode graphique où on est (mode programmé seulement)
MG = 0 (normal)
1 (multicouleurs -résolution 128 sur 64)
2 (haute résolution 128 sur 128 - mais seulement 8 couleurs)
3 (mixte haute résolution et multicouleurs).

C=RCOLR(RC) — donne la couleur contenue dans le registre couleur n° RC
RC=0 (couleur fond d'écran) 1 (couleur du cadre)
RC=2 (couleur caractères) 3 (couleur auxiliaire)
C =0 (noir) 1 (blanc) 2 (rouge) 3 (turquoise)
4 (pourpre) 5 (vert) 6 (bleu) 7 (jaune)
8 (orangé) 9 (orangé clair) 10 (rose)
11(turquoise clair) 12 (pourpre clair)
13(vert clair) 14 (bleu clair) 15(jauneclair)

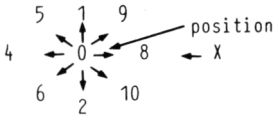
C=RDOT(x,y) — donne la couleur du point de coordonnées x,y. ($0 \leq x,y \leq 1023$)

X=RSND(RS) — donne le contenu du registre sonore n° RS.
RS=1 (voix 1), 2 (voix 2), 3 (voix 3), 4 (bruit) 5 (volume).

X=RPOT(P) — lit la valeur du potentiomètre n°P. P=0 (POT X) ou 1 (POT Y).

X=RPEN(0)
Y=RPEN(1) — lit la position du crayon lumineux.

X=RJOY(Ø) — lit la position de la poignée de jeu
X AND 128 indique l'état du bouton "feu"



KEY n,"texte" — assigne la valeur "texte" à la touche de fonction n° N. Les fonctions préassignées sont : 1 GRAPHIC 2 COLOR, 3 DRAW, 4 SOUND, 5 CIRCLE, 6 POINT, 7 PAINT, 8 LIST C/R.

Commandes graphiques

GRAPHIC MG — sélectionne le mode graphique MG (de 0 à 3)

COLOR E,CD,CAR,A — assigne les valeurs respectives des registres de couleur, dans l'ordre écran, cadre, caractères, auxiliaire.
0 ⩽ E,CD,CAR,A ⩽15

POINT RC,X,Y[,X2,Y2...] — dessine le(s) point(s) de coordonnées indiquées, dans la couleur du registre n° RC (de 0 à 3).

REGION C — échange la couleur des caractères
0⩽C ⩽ 7 en haute résolution
0⩽C ⩽ 15 en multicouleurs

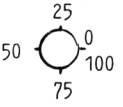
DRAW RC,TO X,Y — trace un trait de couleur contenue dans le registre n° RC du point où on est au point de coordonnées X,Y.

DRAW RC,X,Y TO X1,Y1 — trace un trait de (X,Y) à (X1,Y1).

DRAW RC,X,Y TO X1,Y1 TO X2, Y2 ... — relie par des traits les points (X,Y), (X1,Y1), (X2,Y2) ...

CIRCLE RC,X,Y,RX,RY — trace en couleurs (RC), une ellipse entrée en X,Y et de demi-axes RX et RY. On a un cercle si RY=0.7*RX.

CIRCLE RC,X,Y,RX,RY,D,F — trace un arc du cercle ci-dessus commençant en D et finissant en F, avec l'échelle ci-contre pour D et F.



PAINT RC,X,Y — colore suivant la couleur (RC), l'intérieur de la courbe fermée qui contient le point X,Y.

CHAR ligne, colonne, "texte" — écrit le texte indiqué en commençant en ligne, colonne indiquées. Il faut MG=2 ou 3 et couleur caractères ⩽ 7.

SCNCLR — vide l'écran graphique.

Commandes son

SOUND S1,S2,S3,S4,S5 joue l'accord obtenu en donnant aux 5 registres son les valeurs respectives S1 ... S5 dans l'ordre voix 1, 2, 3, bruit, volume.
0 ⩽S5 ⩽ 15 S5=0=silence
128 < Si <255 = fréquence
Si=0=silence de cette voix

CTRL ← entrée dans le mode musique par touches sur le clavier en mode direct, ou PRINT "ces touches" en programme.

P affiche les caractères sonores en même temps qu'ils sont joués.

Q supprime affichage.

Vv établit le volume 0 ⩽ v ⩽ 15

Sn choisit la voix 1 ⩽n ⩽4

Oi choisit l'octave 0 ⩽i ⩽3

Tt choisit le temps

t	0	1	2	3	4	5	6	7	8	9
durée en tierces	4	6	8	12	16	24	32	64	128	255

C D E F G A B joue la note (resp. DO, RE, MI, FA, SOL, LA SI).

R silence de durée t.

dièze la note qui suit.

$ bémolise la note qui suit.

COMMANDES DU MONITEUR LANGAGE-MACHINE EN CARTOUCHE

Entrée par SYS 6*4096 (SYS 24576)

Commandes principales

.G adr	exécution
.L "nom",pp	chargement
.M adr1,adr2	affichage mémoire
.R	affichage registres
.S "nom",pp,adr1,adr2	sauvegarde
.X	retour à Basic.

Commandes supplémentaires

.A adr instruction	mini-assembleur
.B adr, nb	établit un point d'arrêt en adr pour le nb ie passage.
.D adr1[,adr2]	désassemblage
.E adr	établit une page zéro virtuelle.
.F adr1 adr2 val	remplissage mémoire
.H adr1 adr2 valeurs	recherche en mémoire
.I adr1 adr2	affichage mémoire avec visualisation des codes ASCII
.J adr	exécute un sous-programme à pleine vitesse.
.N adr1,adr2,dépl,adr4,adr5[,W]	corrige (ajoute depl) les adresses absolues comprises entre adr4 et adr5 dans la zone mémoire adr1-adr2. (sert après la commande T)
.Q adr	lance l'exécution avec trace.
.RB adr	supprime le point d'arrêt en adr
.T adr1 adr2 adr3	translation mémoire
.W adr	lance l'exécution en pas-à-pas

CODES CARACTERES DU VIC (EN DECIMAL)

Caractère	*Code écran**	*ASCII*	*Caractère*	*Code écran**	*ASCII*
A	1	65	T	20	84
B	2	66	U	21	85
C	3	67	V	22	86
D	4	68	W	23	87
E	5	69	X	24	88
F	6	70	Y	25	89
G	7	71	Z	26	90
H	8	72			
I	9	73	0	48	48 +
J	10	74	1	49	49 +
K	11	75	2	50	50 +
L	12	76	3	51	51 +
M	13	77	4	52	52 +
N	14	78	5	53	53 +
O	15	79	6	54	54 +
P	16	80	7	55	55 +
Q	17	81	8	56	56 +
R	18	82	9	57	57 +
S	19	83			

CODES CARACTERES DU VIC (EN DECIMAL)

Caractère	*Code écran**	*ASCII*
+	43	43 +
-	45	45 +
*	42	42 +
/	47	47 +
↑	30	94
.	46	46 +
=	61	61 +
<	60	60 +
>	62	62 +
!	33	33 +
?	63	63 +
:	58	58 +
;	59	59 +
,	44	44 +
(	40	40 +
)	41	41 +
"	34	34 +
'	39	39 +
=	35	35 +
$	36	36 +
%	37	37 +
&	38	38 +

Caractère	*Code écran**	*ASCII*
£	28	92
←	31	95
π	94	222
[	27	91
]	29	93
@	0	64
♠	65	193
♥	83	211
♦	90	218
♣	88	216
▗	125	189 +
▄	109	173 +
▖	112	176 +
▝	110	174 +
▟	126	190 +
▙	124	188 +
▜	108	172 +
▛	123	187 +
Sp	32/96	32 +
o	79	207

Caractère	Code écran*	ASCII
p	80	208
v	122	186 +
l	76	204
n	78	206
m	77	205
v	86	214
	91	219
	127	191
w	87	215
q	81	209
	116	180 +
	117	181 +
	97	161 +
	118	182 +
	106	170 +
	105	169 +
	95	223
	102	166 +
	104	168 +
	92	220
	119	183 +
	120	184 +
	98	162 +
	121	185 +
	111	175 +
	114	178 +
	115	179 +
	107	171 +
	113	177 +
j	74	202
k	75	203
u	85	213
i	73	201
	99	163 +
e	69	197
d	68	196
c	67	195
	64	192
f	70	198
r	82	210
	100	164 +

CODES CARACTERES DU VIC (EN DECIMAL)

Caractère	*Code écran**	*ASCII*
	101	165 +
t	84	212
g	71	199
b	66	194
	93	221
h	72	200
y	89	217
	103	167 +
Return		13 +
d		29
g		157
b		17

Caractère	*Code écran**	*ASCII*
h		29
Home		19
Clr		147
Del		20
Inst		148
Rvs^{C}		18
Off^{C}		146
Blk^{C}	0	144
Wht^{C}	1	5
Red^{C}	2	28
Cyn^{C}	3	159
Pur^{C}	4	156
Grn^{C}	5	30
Blu^{C}	6	31
Yel^{C}	7	158

Notes sur les tableaux

Si code ASCII du caractère x égale a, alors code ASCII de 'shift' x égale a+128 (a+80 hex).

+ Lorsqu'un code ASCII, a, est suivi de +, c'est qu'on peut aussi bien employer le code a+64 (a+40 hex).

* Pour obtenir le caractère en contraste inversé, ajouter 128 (80 hex) au code-écran.

S'il y a deux caractères dans une même colonne, le premier correspond au mode "graphique", le second au mode "minuscules".

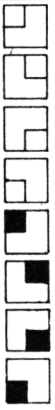
• Code dans la mémoire de couleur.

CODES CARACTERES DU VIC (EN HEXADECIMAL)

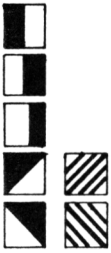
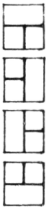
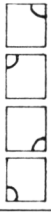
Caractère	*Code écran**	*ASCII*	*Caractère*	*Code écran**	*ASCII*
A	1	41	Y	19	59
B	2	42	Z	1A	5A
C	3	43			
D	4	44	0	30	30 +
E	5	45	1	31	31 +
F	6	46	2	32	32 +
G	7	47	3	33	33 +
H	8	48	4	34	34 +
I	9	49	5	35	35 +
J	A	4A	6	36	36 +
K	B	4B	7	37	37 +
L	C	4C	8	38	38 +
M	D	4D	9	39	39 +
N	E	4E			
O	F	4F	+	2B	2B +
P	10	50	-	2D	2D +
Q	11	51	*	2A	2A +
R	12	52	/	2F	2F +
S	13	53	↑	1E	5E
T	14	54	.	2E	2E +
U	15	55	=	3D	3D +
V	16	56	<	3C	3C +
W	17	57	>	3E	3E +
X	18	58	!	21	21 +

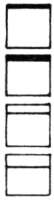
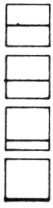
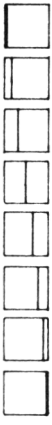
CODES CARACTERES DU VIC (EN HEXADECIMAL)

Caractère	Code écran*	ASCII
?	3F	3F +
:	3A	3A +
;	3B	3B +
,	2C	2C +
(	28	28 +
)	29	29 +
"	22	22 +
.	27	27 +
=	23	23 +
$	24	24 +
%	25	25 +
&	26	26 +
£	1C	5C
←	1F	5F
π	5E	DE
[	1B	5B
]	1D	5D
	0	40
a	41	C1
s	53	D3
z	5A	DA
x	58	D8

Caractère	Code écran*	ASCII
	7D	BD +
	6D	AD +
	70	B0 +
	6E	AE +
	7E	BE +
	7C	BC +
	6C	AC +
	7B	BB +
espace Sp	20/60	20 +
o	4F	CF
p	50	D0
√	7A	BA +
ℓ	4C	CC
n	4E	CE
m	4D	CD
v	56	D6
	5B	DB
	7F	BF
w	57	D7
q	51	D1
	74	B4 +
	75	B5 +

CODES CARACTERES DU VIC (EN HEXADECIMAL)

Caractères	Code écran*	ASCII
	61	A1 +
	76	B6 +
	6A	AA +
	69	A9 +
	5F	DF
	66	A6 +
	6B	AB +
	5C	DC
	77	B7 +
	7B	B8 +
	62	A2 +
	79	B9 +
	6F	AF +
	72	B2 +
	73	B3 +
	6B	AB +
	71	B1 +
 J	4A	CA
k	4B	CB
u	55	D5
i	49	C9

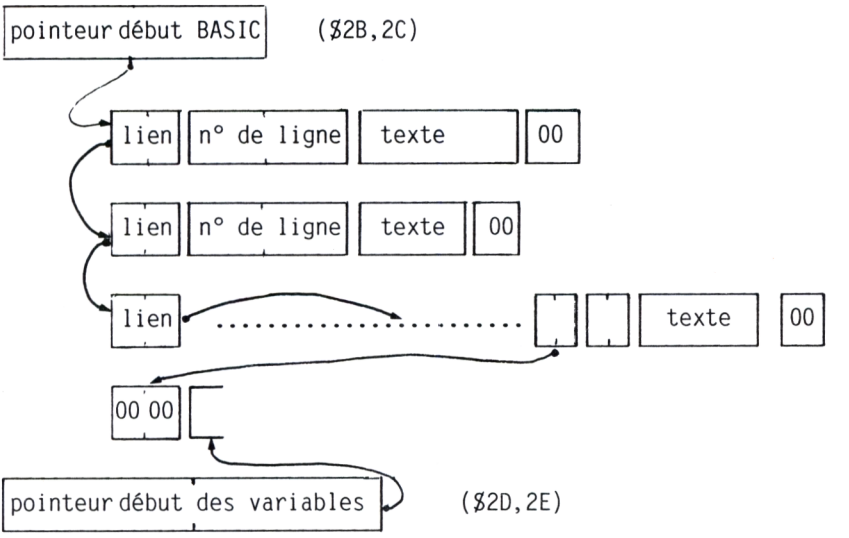
Caractère	Code écran	ASCII
	63	A3 +
e	45	C5
d	44	C4
c	43	C3
	40	C0
f	46	C6
r	52	D2
	64	A4 +
	65	A5 +
t	54	D4
g	47	C7
b	42	C2
	5D	DD
h	48	C8
y	59	D9
	67	A7 +
Return		0D
Return		8D
d		1D
g		9D
b		11

CODES CARACTERES DU VIC (EN HEXADECIMAL)

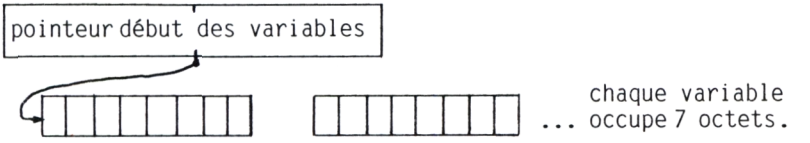
Caractère	*Code écran**	*ASCII*
h		91
Home		13
Clr		93
Del		14
Inst		94
Rvs^{C}		12
Off^{C}		92
Blk^{C}	0•	90
Wht^{C}	1	5
Red^{C}	2	1C
Cyn^{C}	3	9F
Pur^{C}	4	9C
Grn^{C}	5	1E
Blu^{C}	6	1F
Yel^{C}	7	9E

Représentation livre	*Représentation affichage*	*Effet*
'Shift' ou ——	...	Caractère du dessus de la touche
'CTRL' ou C	...	Contrôle
'Space' ou Sp		Espace
'Return'	...	Retour chariot
d		Curseur à droite
g		Curseur à gauche
b		Curseur en bas
h		Curseur en haut
Home		Curseur à l'origine à gauche
Clear ou Clr		Vider l'écran
Rvs C		Inversion du contraste (noir sur blanc)
Off C		Retour au contraste normal (blanc sur noir)
Del		Suppression de caractère
Inst		Insertion de caractère
Blk C		Noir
Wht C		Blanc
Red C		Rouge
Cyn C		Turquoise
Pur C		Pourpre
Grn C		Vert
Blu C		Bleu
Yel C		Jaune

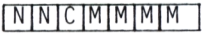
Représentation des Programmes



Organisation des variables simples

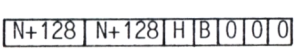


réel



NN codes ASCII des deux premiers caractères du nom
C caractéristique MMMM mantisse

entier

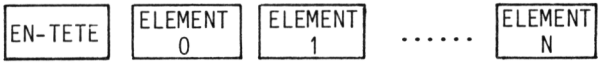


NN on ajoute 128 aux codes ASCII des deux premiers caractères du nom
valeur = 256 * H + B

chaîne de caractères

NN nom (on ajoute 128 au code du 2e car.)
L longueur
PTB + 256 * PTH = adresse mémoire où commence la chaîne proprement dite.

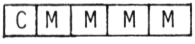
Organisation des tableaux

Tableau = 

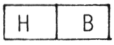
En-tête =

correspond à la déclaration DIM NN (D1,D2,... DK)

NN	deux premiers caractères du nom (+128 selon le type)
PB+256*PH=	adresse de l'en-tête du prochain tableau
K	nombre de dimensions
DK	dernière dimension (ou la seule) sous la forme haut bas
DK-1	avant dernière dimension ...

Elément réel 

C	caractéristique
MMMM	mantisse

Elément entier 

valeur = B+256*H

Elément chaîne de caractères

L = longueur

PTB+256*PTH = adresse où est stockée la chaîne

LA VARIABLE D'ETAT ST

Valeurs de ST en relation avec les cassettes

ST =	*Signification*
0	tout s'est bien passé
4	bloc trop court (dû le plus souvent à confusion entre fichier programme
8	bloc trop long et fichier de données)
16	erreur lecture irréparable
32	erreur sur la somme de contrôle
64	fin de fichier rencontrée
- 128	fin de bande rencontrée

Valeurs de ST en relation avec l'IEEE

ST	*Signification*
1	dépassement de temps en écriture (réponse d'un écouteur après 65 ms).
2	dépassement de temps en lecture (réponse d'un parleur après 65 ms).
64	signal EOI (fin de transmission).
- 128	périphérique non présent (ou non mis sous tension).

Bien entendu, ST = 0 signifie "tout s'est bien passé" (au niveau des échanges sur le bus).

ST	*Signification*
1	erreur de parité.
2	erreur de cadrage (bit d'arrêt manquant).
4	débordement du tampon réception (l'émetteur envoie trop vite).
16	le signal CTS manque*.
64	le signal DSR manque*.
- 128	détection de BREAK*.

* *n'existent pas en interface 3 fils.*

Les messages d'erreur sont de la forme :

?message ERROR in numéro

Précisant le numéro de la ligne où l'erreur a été rencontrée. Le seul cas où le numéro n'est pas précisé est celui d'une commande directe.

Les problèmes les plus courants proviennent d'un mot-clé mal orthographié, d'une virgule manquante ou d'une variable d'un certain type employée à tort etc.

BAD SUBSCRIPT (mauvais indice)

Tentative d'appeler un élément de tableau d'indice supérieur à la limite fournie dans le DIM, ou encore avec un nombre d'indices différent de celui spécifié dans le DIM.
Exemple : DIM A(15) avec A(2Ø) ou A(2,2).

CAN'T CONTINUE (on ne peut pas continuer)

Impossibilité de reprendre l'exécution d'un programme par CONT. C'est le cas s'il y a eu une erreur, ou si, pendant l'interruption, on a modifié ou ajouté une instruction. On peut reprendre par GOTO numéro dans certains cas.

DIVISION BY ZERO (division par zéro)

Vient le plus souvent d'une variable non initialisée.

FORMULA TOO COMPLEX (expression chaîne de caractères trop complexe)

ILLEGAL DIRECT (illégal en mode direct)

INPUT, GET et DEF FN sont interdites en mode direct.

ILLEGAL QUANTITY (valeur erronée)

Emploi d'une fonction avec un argument hors de l'intervalle permis.

Exemples :
- indice négatif ou >32767
- argument de LOG négatif ou nul
- argument de SQR négatif
- 0 puissance nombre négatif
- longueur spécifiée dans MID$, RIGHT$, LEFT$ non comprise entre 0 et 255
- index de ON GOTO hors des limites
- adresses dans PEEK, POKE, WAIT ou SYS non comprises entre 0 et 65535.
- octet spécifié dans WAIT, POKE, TAB, ou SPC non compris entre 0 et 255.

NEXT WITHOUT FOR (NEXT sans FOR correspondant)

Vient le plus souvent de boucles mal imbriquées, d'une confusion sur la variable marquée dans le NEXT ou encore, si l'on a supprimé un FOR pour une correction en oubliant de supprimer le NEXT.

OUT OF DATA (DATA épuisées)

On essaie de faire un READ alors que l'on a déjà "lu" toutes les données de toutes les DATA. Il faut, soit comptabiliser les données avec soin, soit utiliser RESTORE. Noter que ce message apparaît si l'on fait 'Return' sur la ligne du message READY qui est interprété comme READ Y.

OUT OF MEMORY (mémoire épuisée)

Programme trop long, ou trop de variables, ou trop de boucles et GOSUB imbriqués.

OVERFLOW (dépassement de capacité)

Résultat d'un calcul supérieur à 1.7E38. Dans l'autre sens, un résultat inférieur à 2.9E-39 est indiscernable de 0, mais il n'y a pas de message d'erreur.

REDIM'D ARRAY (tableau redimensionné)

On a repassé une deuxième fois sur l'instruction DIM portant sur un tableau, ou il y a une deuxième instruction DIM pour un tableau.

REDO FROM START (recommencez depuis le début)

Lors d'une instruction INPUT, on a fourni une quantité alphanumérique, alors que l'on s'attendait à du numérique. Il faut reprendre, en redonnant toutes les valeurs attendues par l'instruction INPUT considérée.

RETURN WITHOUT GOSUB (retour sans GOSUB)

On est tombé sur un RETURN alors que l'on ne venait pas d'exécuter un GOSUB. Dû le plus souvent à l'oubli de END en fin du programme principal qui précède un sous-programme.

STRING TOO LONG (chaîne de caractères trop longue)

Tentative de créer (par concaténation) une chaîne de plus de 255 caractères.

SYNTAX (erreur de syntaxe)

Instruction incompréhensible pour Basic. Dû notamment à des parenthèses non appariées, caractères illégaux, mauvaise ponctuation, faute d'orthographe dans un mot-clé ...

TYPE MISMATCH (désaccord entre numérique et alphanumérique)

Valeur numérique affectée à une variable chaîne, ou vice-versa, ou bien argument numérique fourni à une fonction qui demande un argument alphanumérique, ou vice-versa.

UNDEF'D STATEMENT (instruction non définie)

GOTO, GOSUB ou THEN renvoyant à un numéro de ligne inexistant, C'est le plus souvent dû à l'oubli de correction d'un GOTO... quand la ligne cible a été supprimée ou changée de place.

UNDEF'D FUNCTION (fonction non définie)

Référence à une fonction pour laquelle on a oublié le DEF FN.

BAD DATA (données incorrectes)

Une donnée alphanumérique est trouvée alors qu'une donnée numérique était attendue. (Même chose que REDO FROM START pour INPUT au clavier, mais, au clavier, il suffit que l'opérateur retape les données, ce qu'il ne peut pas faire pour un autre périphérique, d'où un message différent).

DEVICE NOT PRESENT (périphérique absent)

Essai d'adresser un périphérique IEEE qui n'est pas branché, (ou pas mis sous tension, oubli fréquent) ; ou alors erreur sur le numéro de périphérique attribué. Peut arriver sur OPEN, CLOSE, CMD, INPUT#, GET# ou PRINT#.

FILE NOT FOUND (fichier non trouvé)

FILE NOT OPEN (fichier non ouvert)

Manque l'instruction OPEN, ou bien le fichier a été fermé à la suite d'une erreur ou d'une commande CLR, NEW ou RUN.

FILE OPEN (fichier déjà ouvert)

Essai de rouvrir un fichier alors qu'il n'a pas été fermé.

NOT INPUT FILE (fichier non ouvert en entrée)

Essai de lire sur un fichier ouvert en écriture ou sur un périphérique à écriture seule (ex. l'écran).

NOT OUTPUT FILE (fichier non ouvert en sortie)

Essai d'écrire sur un fichier ouvert en lecture ou sur un périphérique à lecture seule (ex. le clavier).

Pour changer le mode d'utilisation d'un fichier, il faut le fermer puis le rouvrir (plus rebobinage manuel dans le cas des cassettes).

LOAD (erreur au chargement)

On n'arrive pas à lire assez bien la cassette pour charger le programme voulu.

Remède : réessayer, éventuellement après démagnétisation et réalignement des têtes.

MESSAGES D'ERREUR CONCERNANT LES FICHIERS

VERIFY (erreur à la vérification)

Le programme tel qu'il est en mémoire ne concorde pas avec la version qu'on vient de sauver. Refaire la sauvegarde.

BREAK (arrêt)

On a fait STOP alors qu'on avait demandé une opération sur cassette.

Liste sous la forme :

Numéro ***MESSAGE IMPRIME*** traduction [*piste, secteur*]
explications succinctes éventuelles

Messages d'état Ce ne sont pas des erreurs

ØØ ***OK*** tout va bien [*ØØ, ØØ*]

Ø1 ***FILES SCRATCHED*** fichiers effacés [*nb. de fichiers effacés, ØØ*] message obtenu après une commande de suppression.

Erreurs lecture

2Ø ***READ ERROR*** erreur de lecture : tête de bloc non trouvée [*piste et secteur concernés*]
Dû le plus souvent à un numéro de secteur illégal ou à une destruction d'informations sur la disquette.

21 ***READ ERROR*** pas de caractère de synchronisation [*piste, secteur*]
Dû à un mauvais alignement de la tête ou absence de disquette montée dans la platine.

22 ***READ ERROR*** bloc de données absent [*piste, secteur*]
Essai de lire un secteur mal écrit, ou couple piste secteur illégal.

23 ***READ ERROR*** erreur de somme de contrôle dans un bloc de données [*piste, secteur*]
Peut dénoter un mauvais contact de masse.

24 ***READ ERROR*** erreur de décodage d'un octet [*piste, secteur*]
Erreur hardware donnant un motif binaire illégal. Peut dénoter un mauvais contact de masse.

27 ***READ ERROR*** comme 23 mais dans un bloc d'en-tête.

Erreurs d'écriture

25 ***WRITE ERROR*** erreur à la vérification [*piste, secteur*]
Toute écriture est suivie d'une lecture de vérification. Cette erreur indique qu'il y a un désaccord entre les données relues et celles qui ont servi à l'écriture.

26 ***WRITE PROTECT ON*** essai d'écrire sur une disquette en protection écriture [*piste, secteur*]

28 ***WRITE ERROR*** bloc trop long [*piste, secteur*]
Disquette mal formatée ou malfonction hardware.

29 ***DISK ID MISMATCH*** désaccord avec l'ID du disque [*piste, secteur*]
Essai d'écrire sur une disquette non initialisée, ou dont l'en-tête a été détruite.

Erreurs de syntaxe

30 ***SYNTAX ERROR*** erreur de syntaxe générale [*00, 00*]
Nombre de noms de fichiers cités dans une commande illégal (ex. deux noms de fichiers destination dans un COPY).

31 ***SYNTAX ERROR*** commande incorrecte [*00, 00*]
Lettre de commande illégale ou pas en première position dans la chaîne.

32 ***SYNTAX ERROR*** commande de plus de 40 caractères [*00, 00*]

33 ***SYNTAX ERROR*** nom de fichier incorrect [*00, 00*]
Nom abrégé/ambigü utilisé alors que l'on n'est pas en lecture.

34 ***SYNTAX ERROR*** pas de fichier indiqué [*00, 00*]
Le nom de fichier a été oublié ou - cas plus fréquent - il n'est pas reconnu comme tel (manque de guillemets ou de ;).

39 ***SYNTAX ERROR*** pas de fichier commande [*00, 00*]
Commande non reconnaissable.

Erreurs sur les fichiers relatifs

50 ***RECORD NOT PRESENT*** enregistrement absent [*piste, secteur*]
Essai de lire ou de se positionner après la fin de fichier.

51 ***OVERFLOW IN RECORD*** débordement d'enregistrement [*piste, secteur*]
Essai d'écrire un enregistrement plus grand que la taille déclarée.

52 ***FILE TOO LARGE*** fichier trop grand [*piste, secteur*]
Le numéro d'enregistrement va au-delà de la capacité du disque.

Erreurs sur les fichiers

60 ***WRITE FILE OPEN*** fichier déjà ouvert en écriture [*00, 00*]
Oubli de CLOSE avant de rouvrir le fichier en lecture.

61 ***FILE NOT OPEN*** fichier non ouvert [*00, 00*]
Manque OPEN.

62 ***FILE NOT FOUND*** le fichier demandé n'existe pas sur la disquette indiquée [*00, 00*]

63 ***FILE EXISTS*** il existe déjà sur la disquette un fichier de même nom que celui que l'on cherche à créer [*00, 00*]

64 ***FILE TYPE MISMATCH*** désaccord sur le type du fichier [*00, 00*]

65 ***NO BLOCK*** bloc non disponible [*piste et secteur du prochain bloc disponible*]
En liaison avec la commande d'allocation de bloc B-A

66 ***ILLEGAL TRACK OR SECTOR*** piste ou secteur inexistant [*piste, secteur*]

67 ***ILLEGAL SYSTEM TRACK OR SECTOR*** piste ou secteur système illégal [*00, 00*]
Tentative d'accès à un secteur réservé au DOS.

Erreurs système

70 ***NO CHANNEL*** plus de canal disponible [*00, 00*]
Canal demandé déjà ouvert ou dépassement de la limite de 5 canaux en séquentiel, ou 6 en accès direct.

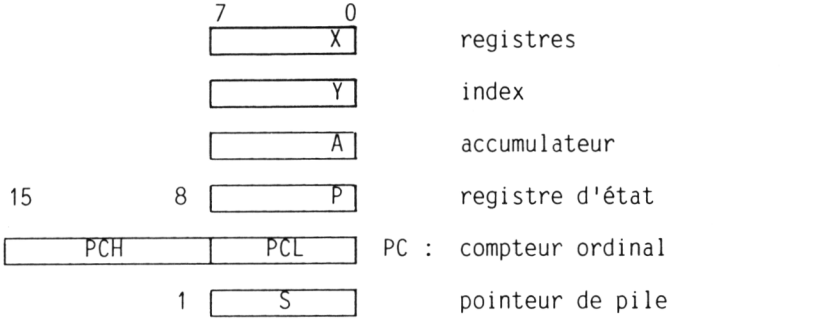
71 ***DIR ERROR*** erreur sur la Table des Secteurs Disponibles (BAM) [*00, 00*]
La TSD a été détruite en mémoire. Il faut réinitialiser la disquette, ce qui peut détruire des fichiers pendants.

72 ***DISK FULL*** disque plein [*00, 00*]
Ou bien tous les blocs sont utilisés ou bien le répertoire arrive à sa limite (152 noms de fichier).

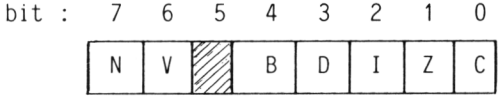
73 ***CBM DOS V2*** désaccord de DOS [*00, 00*]
Essai d'écrire avec une 1540 sur une disquette formatée en 3040.

74 ***DRIVE NOT READY*** platine non prête [*ØØ, ØØ*]
Tentative d'accès alors qu'aucune disquette n'est montée ou demande de la platine 1 : la 1540 ne possède que la platine 0.

REGISTRES INTERNES DU 6502



Détail du registre d'état

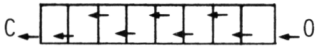


N	signe	D	mode décimal
V	débordement	I	inhibition interruptions
5	inutilisé	Z	résultat nul
B	indicateur de Break	C	retenue

ADC : Addition avec retenue (ADd with Carry) : A← A + M + C
On ajoute à l'accumulateur la mémoire spécifiée plus le bit de retenue.
Opère en mode binaire ou décimal. Agit sur N,V,Z,C.

AND : Et logique (AND) : $A \leftarrow A \wedge M$
Fait le et bit à bit entre accumulateur et mémoire. Agit sur N,Z.

ASL : Décalage à gauche (Arithmétic Shift Left)



Décale à gauche l'accumulateur ou une mémoire. Agit sur N,Z,C.

BCC : Branchement si pas de retenue (Branch on Carry Clear)
Si le bit C = 0, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BCS : Branchement si retenue (Branch on Carry Set)
Si le bit C = 1, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BEQ : Branchement si résultat = 0 (Branch on EQual)
Si le bit Z = 1 (c'est-à-dire si le dernier résultat est 0 ou si la dernière comparaison a donné l'égalité), on saute à l'instruction indiquée ; sinon, on continue en séquence.

BIT : Test de bits (BIt Test) $Z \leftarrow \sum_i \overline{A_i \wedge M_i}$, N ← M7 , V← M6.

Effectue le et virtuel de l'accumulateur et de la mémoire spécifiée et positionne Z en conséquence. En outre, les bits 7 et 6 de la mémoire sont copiés respectivement dans N et V.

BMI : Branchement si négatif (Branch on MInus).
Si le bit N = 1, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BNE : Branchement si non égal à 0 (Branch on Not Equal)
Si le bit Z = 0 (c'est-à-dire si le dernier résultat est ≠ 0 ou si la dernière comparaison n'a pas donné l'égalité), on saute à l'instruction indiquée ; sinon, on continue en séquence.

BPL : Branchement si positif ou nul (Branch if Plus)
Si le bit N = 0, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BRK : Interruption software (BReaK)
Met le bit B à 1 et simule une interruption (saut à l'adresse contenue en FFFE, FFFF).

BVC : Branchement si pas de débordement (Branch on oVerflow Clear).
Si le bit V = 0, on saute à l'instruction indiquée ; sinon, on continue en séquence.

BVS : Branchement si débordement (Branch on oVerflow Set)
Si le bit V = 1, on saute à l'instruction indiquée ; sinon, on continue en séquence.

CLC : Annuler la retenue (CLear Carry)
Force à 0 le bit de retenue.

CLD : Annuler le mode décimal (CLear Decimal mode)
Force à 0 le bit D pour mettre l'UAL en mode binaire.

CLI : Autorise les interruptions (CLear Interrupt inhibit flag)
Force à 0 le bit I d'inhibition des interruptions.

CLV : Annuler l'indicateur de débordement (CLear oVerflow flag)
Force à 0 le bit V.

CMP : Comparer avec l'accumulateur (CoMPare accumulator) A-M
Effectue la soustraction virtuelle registre A - mémoire et positionne les indicateurs N,Z et C en conséquence.

CPX : Comparer avec X (ComPare with X) X-M
Effectue la soustraction virtuelle registre X - mémoire et positionne les indicateurs N,Z et C en conséquence.

CPY : Comparer avec Y (ComPare with Y) Y-M
Effectue la soustraction virtuelle registre Y - mémoire et positionne les indicateurs N, Z et C en conséquence.

DEC : Décrémenter en mémoire (DECrement memory) M←M-1.
Diminue de 1 le contenu de la mémoire indiquée. Agit sur N et Z.

DEX : Décrémenter X (DEcrement X) X←X-1.
Diminue de 1 le contenu du registre X. Agit sur N et Z.

DEY : Décrémenter Y (DEcrement Y) Y←Y-1.
Diminue de 1 le contenu du registre Y. Agit sur N et Z.

EOR : Ou exclusif (Exclusive OR)
Effectue le ou exclusif entre l'accumulateur et la mémoire indiquée. Agit sur N et Z.

INC : INCrémenter en mémoire (Increment memory) M←M+1
Augmente de 1 le contenu de la mémoire indiquée. Agit sur N et Z.

INX : INcrémenter X (Increment X) X←X+1
Augmente de 1 le contenu du registre X. Agit sur N et Z.

INY : INcrémenter Y (Incrément Y) Y←Y+1
Augmente de 1 le contenu du registre Y. Agit sur N et Z.

JMP : Saut inconditionnel (JuMP) PC ← adresse
Saute à l'adresse indiquée.

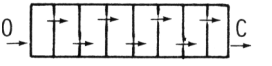
JSR : Appel d'un sous-programme (Jump to Sub Routine) PC ↓ ;
PC ← Ad
Sauve PC dans la pile (adresse de retour) puis saute à l'adresse indiquée.

LDA : Charger l'accumulateur (LoaD Accumulator) A ← M
Met dans l'accumulateur le contenu de la mémoire spécifiée. Agit sur N et Z.

LDX : Charger X (LoaD X register) X ← M
Met dans le registre X le contenu de la mémoire spécifiée. Agit sur N et Z.

LDY : Charger Y (LoaD Y register) Y ← M
Met dans le registre Y le contenu de la mémoire spécifiée. Agit sur N et Z.

LSR : Décalage à droite (Logical Shift Right)
Décale à droite l'accumulateur ou une mémoire. Agit sur N, Z et C.



NOP : Pas d'opération (No OPeration) PC←PC+1
Instruction muette. S'exécute en 2 cycles.

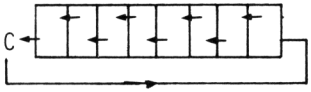
ORA : Ou inclusif (OR Accumulator) A←AvM
Effectue le ou inclusif entre l'accumulateur et la mémoire indiquée. Agit sur N et Z.

PHA : Empiler A (PusH Accumulator) A ↓ : (S) ← A ; S ← S - 1
Met l'accumulateur au sommet de la pile et met à jour le pointeur de pile.

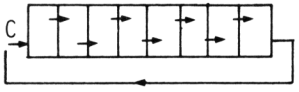
PHP : Empiler P (PusH Processor status register)
P ↓ : (S) ← P ; S ← S - 1
Met le registre d'état P au sommet de la pile et met à jour le pointeur de pile.

PLA : Dépiler vers A (PuLl Accumulator) A ↑ : S ← S + 1;A← (S)
Transfére vers le registre d'état P le contenu du sommet de la pile et met à jour le pointeur de pile.
Affecte tous les indicateurs.

ROL : Rotation à gauche (ROtate Left)
Décale à gauche l'accumulateur ou une mémoire. L'ancienne valeur du bit de retenue rentre par la droite tandis que le bit qui sort par la droite vient remplacer C. Affecte N, Z et C.



ROR : Rotation à droite (ROtate Right)
Décale à droite l'accumulateur ou une mémoire. L'ancienne valeur du bit de retenue rentre par la gauche tandis que le bit qui sort par la droite vient remplacer C. Affecte N, Z et C.



RTI : Retour d'interruption (ReTurn from Interrupt)
P ↑ ; PC ↑
Retour de routine d'interruption : récupère sur la pile PC et P qui y avaient été sauvés par le mécanisme d'interruption.

RTS : Retour du sous-programme (ReTurn from Subroutine) PC ↑
Récupère sur la pile PC qui y avait été sauvé par le dernier JSR.

SBC : Soustraction avec retenue (SuBtract with Carry)
A ← A - M - $\overline{C}$
On soustrait à l'accumulateur la mémoire spécifiée et aussi l'opposé du bit de retenue (emprunt). Opère en mode binaire ou décimal. Agit sur N, V, Z, C.

SEC : Mettre à 1 la retenue (SEt Carry flag)
Force à 1 le bit C.

SED : Mettre en mode décimal (SEt Decimal mode)
Force à 1 le bit D (influe sur ADC et SBC).

SEI : Inhiber les interruptions (SEt Interrupt flag)
Force à 1 le bit I.

STA : Ranger l'accumulateur (STore Accumulator) M ← A
Transfère le contenu de l'accumulateur dans la mémoire indiquée.

STX : Ranger X (STore X register) M ← X
Transfère le contenu du registre X dans la mémoire indiquée.

STY : Ranger Y (STore Y register) M ← Y
Transfère le contenu du registre Y dans la mémoire indiquée.

TAX : Transfert de A dans X. X ← A. Agit sur N et Z.

TAY : Transfert de A dans Y. Y ← A. Agit sur N et Z.

TSX : Transfert de S dans X. X ← S. Agit sur N et Z.

TXA : Transfert de X dans S. A ← X. Agit sur N et Z.

TXS : Transfert de X dans S. S ← X. N'agit pas sur les indicateurs.

TYA : Transfert de Y dans A. A ← Y. Agit sur N et Z.

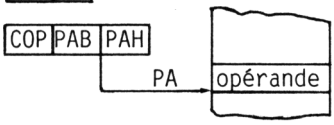
COP = code opération : PA = partie adresse
AE = adresse effective : () = contenu de

Inhérent — COP — adresse implicite

Accumulateur — COP — opérande=accumulateur

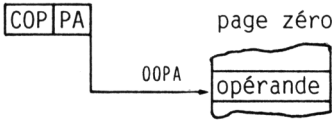
Immédiat — COP | PA — opérande=PA elle-même

Absolu



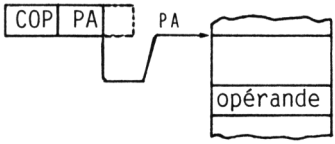
AE=PAH PAB
opérande=(PA)

Page zéro



page zéro AE=00PA

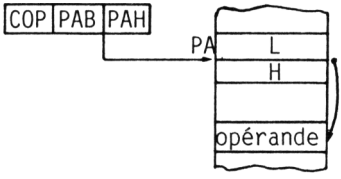
Indexé



AE=PA + (index)

se subdivise en 4 modes
- absolu indexé par X
- absolu indexé par Y
- page zéro indexé par X
- page zéro indexé par Y

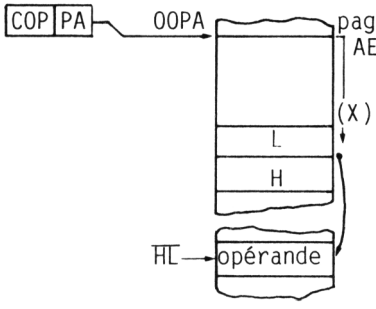
Indirect



PA=adresse de l'adresse
AE=(PA)
opérande=((PA))

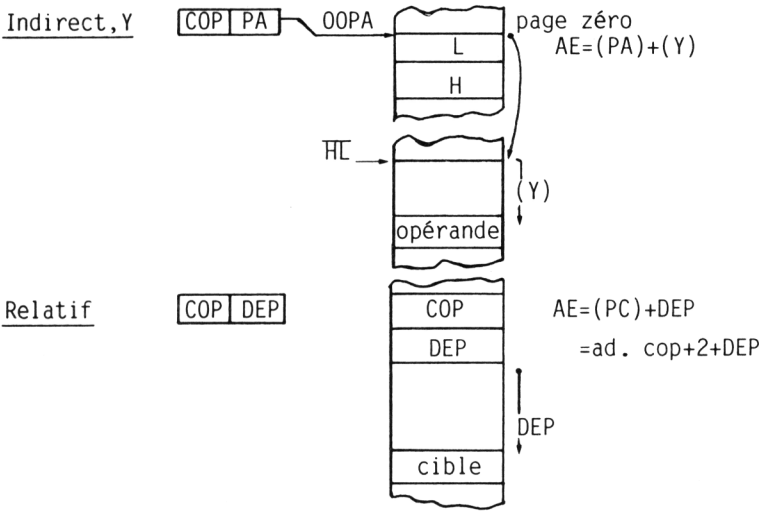
(n'existe que pour JMP)

Indirect,X



AE=(PA+(X))

SCHEMA DES MODES D'ADRESSAGE



Ce tableau est l'inverse du suivant. En fonction du code hexadécimal AB, il donne le mnémonique et le mode d'adressage correspondant. Exemple : A9 → LDA IMM (ligne A, col 9). (Pas de mode d'adressage = inhérent ou relatif).

A \ B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK	ORA IND,X				ORA PZ,X	ASL PZ,X		PHP	ORA IMM	ASL A			ORA ABS	ASL ABS	
1	BPL	ORA IND,Y				ORA PZ,X	ASL PZ,X		CLC	ORA ABS,Y				ORA ABS,X	ASL ABS,X	
2	JSR	AND IND,X			BIT PGE Z	AND PGE Z	ROL PGE Z		PLP	AND IMM	ROL A		BIT ABS	AND ABS	ROL ABS	
3	BMI	AND IND,Y				AND PZ,X	ROL PZ,X		SEC	AND ABS,Y				AND ABS,X	ROL ABS,X	
4	RTI	EOR IND,X				EOR PGE Z	LSR PGE,Z		PHA	EOR IMM	LSR A		JMP ABS	EOR ABS	LSR ABS	
5	BVC	EOR IND,Y				EOR PZ,X	LSR PZ,X		CLI	EOR ABS,Y				EOR ABS,X	LSR ABS,X	
6	RTS	ADC IND,X				ADC PGE Z	ROR PGE Z		PLA	ADC IMM	ROR A		JMP IND	ADC ABS	ROR ABS	
7	BVS	ADC IND,Y				ADC PZ,X	ROR PZ,X		SEI	ADC ABS,Y				ADC ABS,X	ROR ABS,X	
8		STA IND,X			STY PGE Z	STA PGE Z	STX PGE Z		DEY		TXA		STY ABS	STA ABS	STX ABS	
9	BCC	STA IND,Y			STY PZ,X	STA PZ,X	STX PZ,Y		TYA	STA ABS,Y	TXS			STA ABS,X		
A	LDY IMM	LDA IND,X	LDX IMM		LDY PGE Z	LDA PGE Z	LDX PGE Z		TAY	LDA IMM	TAX		LDY ABS	LDA ABS	LDX ABS	
B	BCS	LDA IND,Y			LDY PZ,X	LDA PZ,X	LDX PZ,Y		CLV	LDA ABS,Y	TSX		LDY ABS,X	LDA ABS,X	LDX ABS,Y	
C	CPY IMM	CMP IND,X			CPY PGE Z	CMP PGE Z	DEC PGE Z		INY	CMP IMM	DEX		CPY ABS	CMP ABS	DEC ABS	
D	BNE	CMP IND,Y				CMP PZ,X	DEC PZ,X		CLD	CMP ABS,Y				CMP ABS,X	DEC ABS,X	
E	CPX IMM	SBC IND,X			CPX PGE Z	SBC PGE Z	INC PGE Z		INX	SBC IMM	NOP		CPX ABS	SBC ABS	INC ABS	
F	BEQ	SBC IND,Y				SBC PZ,X	INC PZ,X		SED	SBC ABS,Y				SBC ABS,X	INC ABS,X	

Codes opération/nombre de cycles (n)/nombre d'octets occupés (#)

MNEMO.	EFFET	IMMEDIAT OP	IMMEDIAT #	ABSOLU OP	ABSOLU n	ABSOLU #	PAGE ZERO OP	PAGE ZERO n	PAGE ZERO #	ACCUM. OP	ACCUM. n	ACCUM. #	INHERENT OP	INHERENT n	INHERENT #	(IND,X) OP	(IND,X) n	(IND,X) #	(IND),Y) OP	(IND),Y) n	(IND),Y) #	PAGE Z,X OP	PAGE Z,X n	PAGE Z,X #	ABS,X OP	ABS,X n	ABS,X #	ABS,Y OP	ABS,Y n	ABS,Y #	RELATIF OP	RELATIF n	RELATIF #	INDIRECT OP	INDIRECT n	INDIRECT #	PAGE Z,Y OP	PAGE Z,Y n	PAGE Z,Y #	REGISTRE D'ETAT 7 6 5 4 3 2 1 0 N V . B D I Z C	MNEMO.
A D C	A+M+C → A (1) (4)	69	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										N V . . . . . Z C	A D C
A N D	A∧M → A (1)	29	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										N . . . . . Z .	A N D
A S L	C ← 7 6 5 4 3 2 1 0 ← 0			0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													N . . . . . Z C	A S L
B C C	BRANCH.SI C=0 (2)																														90	2	2								B C C
B C S	BRANCH.SI C=1 (2)																														B0	2	2								B C S
B E Q	BRANCH.SI Z=1 (2)			2C	4	3	24	3	2																						F0	2	2								B E Q
B I T	A∧M																																							M7 M6 . . . . . Z .	B I T
B M I	BRANCH.SI N=1 (2)																														30	2	2								B M I
B N E	BRANCH.SI Z=0 (2)																														D0	2	2								B N E
B P L	BRANCH.SI N=0 (2)																														10	2	2								B P L
B R K	BREAK												00	7	1																										. . . 1 . 1 . . .	B R K
B V C	BRANCH.SI V=0 (2)																														50	2	2								B V C
B V S	BRANCH.SI V=1 (2)																														70	2	2								B V S
C L C	0 → C												18	2	1																										 0	C L C
C L D	0 → D												D8	2	1																										. . . . . 0 . .	C L D
C L I	0 → I												58	2	1																										. . . . . 0 . .	C L I
C L V	0 → V												B8	2	1																										. 0	C L V
C M P	A − M	C9	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										N . . . . . Z C	C M P
C P X	X − M	E0	2	EC	4	3	E4	3	2																															N . . . . . Z C	C P X
C P Y	Y − M	C0	2	CC	4	3	C4	3	2																															N . . . . . Z C	C P Y
D E C	M − 1 → M			CE	6	3	C6	5	2													D6	6	2	DE	7	3													N . . . . . Z .	D E C
D E X	X − 1 → X												CA	2	1																										N . . . . . Z .	D E X
D E Y	Y − 1 → Y												88	2	1																										N . . . . . Z .	D E Y
E O R	A ● M → A (1)	49	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										N . . . . . Z .	E O R
I N C	M+1 → M			EE	6	3	E6	5	2													F6	6	2	FE	7	3													N . . . . . Z .	I N C
I N X	X+1 → X												E8	2	1																										N . . . . . Z .	I N X
I N Y	Y+1 → Y												C8	2	1																										N . . . . . Z .	I N Y
J M P	SAUT INCOND.			4C	3	3																													6C	5	3					. . . . .	J M P
J S R	APPEL S/P			20	6	3																																			. . . . .	J S R
L D A	M → A (1)	A9	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										N . . . . . Z .	L D A

JEU D'INSTRUCTIONS DU 6502

Codes opération/nombre de cycles (n)/nombre d'octets occupés (#)

INSTRUCTIONS		IMMEDIAT			ABSOLU			PAGE ZERO			ACCUM.			INHERENT			(IND,X)			(IND),Y			PAGE Z,X			ABS,X			ABS,Y			RELATIF			INDIRECT			PAGE Z,Y			REGISTRE D'ETAT	
MNEMO	EFFET	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	7 6 5 4 3 2 1 0 N V . B D I Z C	MNEMO
LDX	M→X (1)	A2	2	2	AE	4	3	A6	3	2																BE	4	3										B6	4	2	N . . . . . Z .	LDX
LDY	M→Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													N . . . . . Z .	LDY
LSR	0 → 7 0 → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0 . . . . . Z C	LSR
NOP	PAS D'OPERATION													EA	2	1																									. . . . .	NOP
ORA	AVM→A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										N . . . . . Z .	ORA
PHA	A→Ms S-1→S													48	3	1																									. . . . .	PHA
PHP	P→Ms S-1→S													08	3	1																									. . . . .	PHP
PLA	S+1→S Ms→A													68	4	1																									N . . . . . Z .	PLA
PLP	S+1→S Ms→P													28	4	1																									(RESTAURES)	PLP
ROL	7 0 C				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													N . . . . . Z C	ROL
ROR	C → 7 0				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													N . . . . . Z C	ROR
RTI	RETOUR INTER													40	6	1																									(RESTAURES)	RTI
RTS	RETOUR S/P													60	6	1																									. . . . .	RTS
SBC	A - M - C→A (1) (4	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										N V . . . . Z (3)	SBC
SEC	1→C													38	2	1																									 1	SEC
SED	1→D													F8	2	1																									. . . . 1 . . .	SFD
SEI	1→D													78	2	1																									. . . . . 1 . .	SEI
STA	A→M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3										. . . . .	STA
STX	X→M				8E	4	3	86	3	2																															. . . . .	STX
STY	Y→M				8C	4	3	84	3	2													94	4	2																	. . . . .	STY
TAX	1→X													AA	2	1																									N . . . . . Z .	TAX
TAY	A→Y													A8	2	1																									N . . . . . Z .	TAY
TSX	S→X													BA	2	1																									N . . . . . Z .	TSX
TXA	X→A													8A	2	1																									N . . . . . Z .	TXA
TXS	X→S													9A	2	1																									. . . . .	TXS
TYA	Y→A													98	2	1																									N . . . . . Z .	TYA

(1) Ajouter 1 à "n" si changement de page

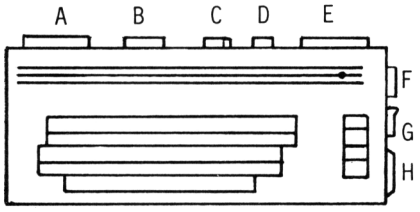
(2) Ajouter 1 à "n" si branchement dans une page différente

(3) Emprunt = $\overline{\text{retenue}}$

(4) En mode décimal, l'indicateur Z est inopérant. L'accumulateur doit être testé pour savoir si le résultat est nul.

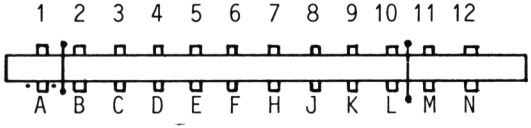
X	Index X	+	Addition
Y	Index Y	-	Soustraction
A	Accumulateur	^	Et
M	Mémoire	v	Ou
Ms	Mémoire (pile)	@	Ou exclusif
M_7	Mémoire (bit 7)	n	Nombre de cycles
M_6	Mémoire (bit 6)	#	Nombre d'octets

JEAN-MICHEL MAGNAN 2056



A. Connecteur "port utilisateur".
B. Connecteur "magnétophone".
C. Connecteur "bus IEEE-série"
D. Connecteur "vidéo".
E. Connecteur "extensions".
F. Connecteur "alimentation".
G. Bouton "marche/arrêt".
H. Connecteur "jeux".

Connecteur "port utilisateur" et RS232

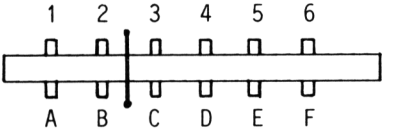


(vue depuis l'arrière du VIC)

= détrompeur

Broche	*Rôle*	*Broche*	*Rôle*	*Rôle RS232*
1	masse	A	masse	masse
2	+5V	B	CB1	Sin
3	RESET	C	PB0	Sin
4	JOY0	D	PB1	RTS
5	JOY1	E	PB2	DTR
6	JOY2	F	PB3	RI
7	cr. Lumin.	H	PB4	DCD
8	inter K7	J	PB5	
9	entrée ATN	K	PB6	CTS
10	9V	L	PB7	DSR
11	altern.	M	CB2	Sout
12	masse	N	masse	masse

Connecteur magnétophone



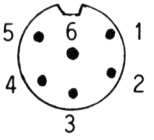
(vue arrière du VIC)

= *détrompeur*

Broche	*Rôle*
1 - A	masse
2 - B	+ 5V
3 - C	moteur
4 - D	lecture
5 - E	écriture
6 - F	test bouton

Connecteur bus IEEE série

Connecteur : DIN 6 broches

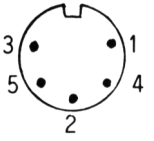


(vue arrière du VIC)

Broche	*Rôle*
1	SRQ
2	masse
3	ATN
4	horloge
5	donnée
6	reset

Connecteur vidéo

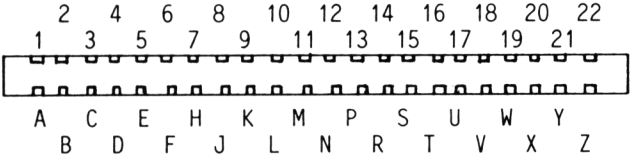
Connecteur : DIN 5 broches



(vue arrière du VIC)

Broche	*Rôle*
1	+6V
2	masse
3	son
4	vidéo basse
5	vidéo haute

Connecteur d'extensions pour cartouches

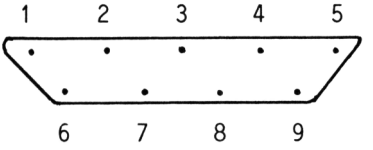


(vue arrière du VIC)

Broche	Rôle	
1	masse	
2	D0	Bus de données
3	D1	
4	D2	
5	D3	
6	D4	
7	D5	
8	D6	
9	D7	
10	$\overline{BLK1}$	sélection $2000-3FFF
11	$\overline{BLK2}$	sélection $4000-5FFF
12	$\overline{BLK3}$	sélection $6000-7FFF
13	$\overline{BLK5}$	sélection $A000-BFFF
14	$\overline{RAM1}$	extension $0400-07FF
15	$\overline{RAM2}$	3K $0800-0BFF
16	$\overline{RAM3}$	RAM $0C00-0FFF
17	VRW	lecture/écriture 6560
18	CRW	lecture/écriture 6502
19	$\overline{IRQ}$	
20	nc	
21	+5V	
22	masse	

Broche	Rôle	
A	masse	
B	A0	Bus d'adresses
C	A1	
D	A2	
E	A3	
F	A4	
H	A5	
J	A6	
K	A7	
L	A8	
M	A9	
N	A10	
P	A11	
R	A12	
S	A13	
T	$\overline{I/O2}$	sélection $9800
U	$\overline{I/O3}$	d'entrée-sortie $9C00
V	SØ2	horloge système Ø2
W	$\overline{NMI}$	
X	$\overline{reset}$	
Y	nc	
Z	masse	

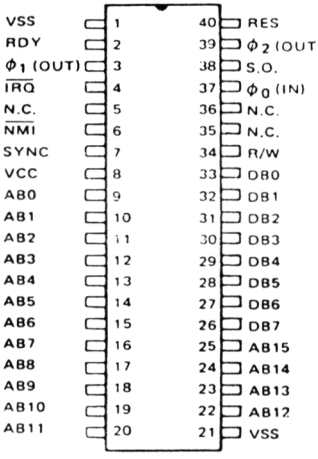
Connecteur de jeux



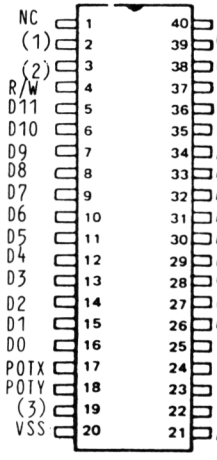
(vue latérale du VIC)

Broche	*Rôle*
1	JOY 0
2	JOY 1
3	JOY 2
4	JOY 3
5	POT Y
6	crayon lumineux
7	+5V
8	masse
9	POT X

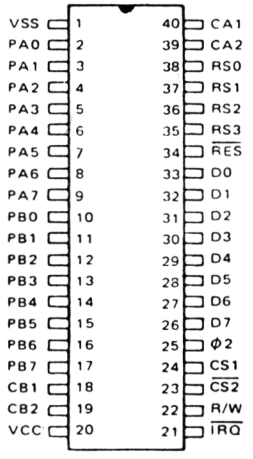
Les broches JOY permettent de connecter des poignées de jeux.



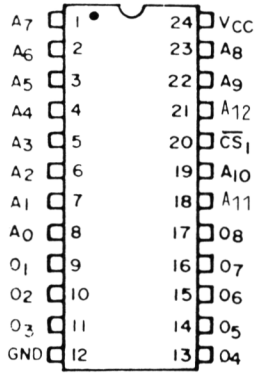
Microprocesseur 6502



VIC 6561

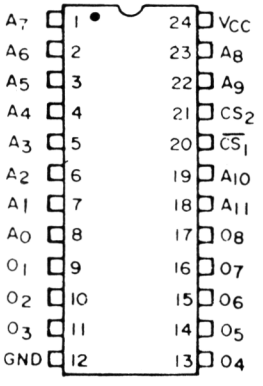


VIA 6522

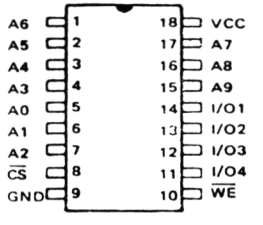


MEM 2364

(1) COMP COLOR
(2) SYNC LUMIN
(3) COMP SOUND
(4) LIGHT PEN



MEM 2332



MEV 2114

Registres des boîtiers d'entrée-sortie

Le boîtier ***VIC 6560***

HEXA				DEC
9000	ENTRELA-CEMENT	MARGE GAUCHE		36864
9001		MARGE HAUTE		36865
9002	AD.ECRAN BIT 9	NOMBRE DE COLONNES		36866
9003	AD.BALAY BIT 0	NOMBRE DE RANGEES	CARACT. DOUBLES	36867
9004	ADRESSE OU EN EST LE BALAYAGE BIT 8 A BIT 1			36868
9005	ADRESSE MEMOIRE ECRAN BITS 13 A 10	ADRESSE GENE. CAR. BITS 13 A 10		36869
9006	CRAYON LUMINEUX HORIZONTAL			36870
9007	CRAYON LUMINEUX VERTICAL			36871
9008	POTENTIOMETRE X			36872
9009	POTENTIOMETRE Y			36873
900A	VOIX 1			36874
900B	VOIX 2			36875

Registres des boftiers d'entrée-sortie (suite)

HEXA		DEC
900C	VOIX 3	36876
900D	VOIX 4 (BRUIT)	36877
900E	MODE MULTI - COULEURS / AMPLITUDE SON	36878
900F	COULEUR FOND ECRAN / CONTRASTE INVERSE / COULEUR CADRE	36879

Registres des boîtiers d'entrée-sortie

*Le **6522 N° 1** (interruption NMI)*

HEXA		DEC
9110	PB — PORT PARALLELE UTILISATEUR ET RS232	37136
9111	PA — UTILISE PAR L'ADRESSE 911F	37137
9112	DIRECTION PB	37138
9113	DIRECTION PA	37139
9114	TEMPORISATEUR 1 T1-CL	37140
9115	T1-CH — VITESSE EMISSION RS232	37141
9116	T1-LL — SYNCHRO. ECRITURE CASSETTE	37142
9117	T1-LH	37143
9118	TEMPORISATEUR 2 T2-L	37144
9119	T2-H — SYNCHRO. ENTREE RS232	37145

Registres des boîtiers d'entrée-sortie (suite)

HEXA									DEC
911A	SR REGISTRE A DECALAGE (INUTILISE)								37146
911B	ACR COM.	T1	COM. T2	COMMANDE SR			VERROUILLAGE PB	PA	37147
911C	PCR CONTROLE EMISSION	CB2 RS232		CONTROLE CB1	CONTROLE MOTEUR	CA2 CASSETTE		CONTROLE CA1	37148
911D	IFR NMI	T1	T2	CB1 ENTREE RS232	CB2	SR	CA1 TOUCHE RESTORE	CA2	37149
911E	IER	T1	T2	CB1	CB2.	SR	CA1	CA2	37150
911F	PA SORTIE ATN	TESTE CASSETTE	CRAYON LUMINEUX	MANCHES A BALAI : JOY2	JOY1	JOY0	CRAYON LUMINEUX	ENTREES SERIE IEEE DONNEE	HORLOGE	37151

Registres des boîtiers d'entrée-sortie

*Le **6522 N° 2** (interruption IRQ)*

HEXA		DEC
9120	PB * SELECTION RANGEE CLAVIER +	37152
9121	PA ENTREE COLONNE CLAVIER	37153
9122	DIRECTION PB	37154
9123	DIRECTION PA	37155
9124	T1-CL SYNCHRO. LECTURE CASSETTE ;	37156
9125	T1-CH RYTHME	37157
9126	T1-LL INTERRUPTIONS CLAVIER	37158
9127	T1-LH	37159
9128	T2-L TIMEOUT BUS IEEE SERIE	37160
9129	T2-H SYNCHRO. L/E CASSETTE	37161

Registres des boîtiers d'entrée-sortie

HEXA									DEC
912A	SR REGISTRE A DECALAGE (INUTILISE)								37162
912B	ACR COM. T1		COM. T2	COMMANDE S R			VERROUILLAGE PB	PA	37163
912C	PCR CONTROLE CB2 SORTIE DONNEE IEEE			CONTROLE CB1	CONTROLE CA2 SORTIE HORLOGE IEEE			CONTROLE CA1	37164
912D	IFR IRQ	T1	T2	CB1	CB2	SR	CA1	CA2	37165
912E	IER	T1	T2	CB1	CB2	SR	CASSETTE	CA2	37166
912F	PA UTILISE PAR L'ADRESSE 9121								37167

** Le bit 7 sert aussi d'entrée JOY 3*
+ Le bit 3 sert aussi de sortie écriture cassette.

RESET : 64802 $ FD22
RESET tiède : 50292 $ C474
MONITEUR LANGAGE MACHINE : 24576 $ 6000 (cartouche).

Routines fondamentales

$ FFD2 Impression d'un caractère (dont le code est en A).
$ FF70 Test de la touche STOP.

Une routine utile est la routine CHRGET qui prend dans l'accumulateur un caractère dans le texte Basic. CHRGET est en MEV en $0073. Le caractère à prendre est pointé par $7A,$7B. Un autre point d'entrée est $79 où, comme le pointeur n'est pas incrémenté, on reprend le caractère qu'on avait déjà pris. La routine passe les blancs et s'arrête aux deux points.

Tables de traduction des emplacements clavier en code ASCII

	Touche naturelle	*Avec SHIFT*	*Avec 'C='*	*Avec CTRL*
Hexa	EC5E	EC9F	ECE0	EDA3
Dec	60510	60575	60640	60835

Sortie d'interruption

En $EB18 se trouve une séquence de sortie d'interruption qui rétablit les registres tels qu'ils avaient été sauvegardés.

ADRESSES DES ROUTINES-SYSTEME

N.B. On indique les points de début des routines. Ce ne sont pas forcément les points d'entrée.

HEXA	*DEC*	*Rôle*
C000	49152	vrs (vecteurs) ROM
C00C	49164	vrs action des mots-clés
C052	49234	vrs fonctions
C080	59280	vrs opérations
C09E	49310	mots clés
C19E	49566	messages d'erreur
C328	49960	vrs messages d'erreur
C365	50021	messages divers
C38A	50058	balaie la pile pour FOR / GOSUB
C3B8	50104	translation en mémoire
C3FB	50171	vérifie profondeur pile
C408	50184	vérifie espace mémoire
C435	50229	"OUT OF MEMORY"
C437	50231	routine d'erreur
C469	50281	entrée BRK
C474	50292	"READY"
C480	50304	préparation pour basic
C49C	50332	traite nouvelle ligne
C533	50483	rétablit le chaînage
C560	50528	reçoit ligne d'entrée
C579	50553	remplace les mots clés par leur code
C613	50707	trouve ligne Basic
C642	50754	effectue NEW
C65E	50782	effectue CLR
C68E	50830	copie pointeur texte
C69C	50844	effectue LIST
C742	51010	effectue FOR
C7ED	51181	effectue instruction
C81D	51229	effectue RESTORE
C82C	51244	arrêt
C82F	51247	effectue STOP
C831	51249	effectue END
C857	51287	effectue CONT
C871	51313	effectue RUN
C883	51331	effectue GOSUB
C8A0	51360	effectue GOTO
C8D2	51410	effectue RETURN
C8F8	51448	effectue DATA
C906	51462	recherche prochaine instruction
C928	51496	effectue IF
C93B	51515	effectue REM
C94B	51531	effectue ON
C96B	51563	prend un nombre entier

HEXA	*DEC*	*Rôle*
C9A5	51621	effectue LET
CA80	51840	effectue PRINT#
CA86	51846	effectue CMD
CAA0	51872	effectue PRINT
CB1E	51998	imprime message pointé par (Y,A)
CB3B	52027	imprime caractère de format
CB4D	52045	routine mauvaise entrée
CB7B	52091	effectue GET
CBA5	52133	effectue INPUT#
CBBF	52159	effectue INPUT
CBF9	52217	interroge pour INPUT
CC06	52230	effectue READ
CCFC	52476	message d'erreur en entrée
CD1E	52510	effectue NEXT
CD78	52600	vérifie accord de type
CD9E	52638	effectue expression
CEA8	52904	constante "PI"
CEF1	52977	effectue les parenthèses
CEF7	52983	vérifie ")"
CEFF	52991	vérifie ","
CF08	53000	erreur de syntaxe
CF14	53012	vérifie domaine
CF28	53032	cherche variable
CFA7	53159	référence à FN...
CFE6	53222	effectue OR
CFE9	53225	effectue AND
D016	53270	comparaison
D081	53377	effectue DIM
D08B	53387	localise variable
D113	53523	vérifie caractère alphabétique
D11D	53533	crée une variable
D194	53652	sp pointeur vers tableau
D1A5	53669	valeur 32768
D1B2	53682	convers. flottant/entier
D1D1	53713	prépare tableau
D245	53829	"BAD SUBSCRIPT"
D248	53832	"ILLEGAL QUANTITY"
D34C	54092	calcule taille tableau
D37D	54141	effectue FRE
D391	54161	convers. entier/flottant
D39E	54174	effectue POS
D3A6	54182	vérifie mode direct
D3B3	54195	effectue DEF
D3E1	54241	vérifie syntaxe de FN
D465	54373	effectue STR$
D475	54389	calcule vect de chaîne
D487	54407	prépare chaîne

ADRESSES DES ROUTINES-SYSTEME

HEXA	*DEC*	*Rôle*
D4F4	54516	fait de la place pour une chaîne
D526	54566	"garbage collection"
D5BD	54717	teste si à conserver
D606	54790	rassemble chaîne
D63D	54845	concaténation
D67A	54906	construit chaîne en mémoire
D6A3	54947	rejette chaîne
D6DB	55003	nettoie la pile des descripteurs
D6EC	55020	effectue CHR$
D700	55040	effectue LEFT$
D72C	55084	effectue RIGHT$
D737	55095	effectue MID$
D761	55137	dépile les paramètres de chaîne
D77C	55164	effectue LEN
D782	55170	sort du mode chaîne
D78B	55179	effectue ASC
D79B	55195	obtient octet paramètre
D7AD	55213	effectue VAL
D7EB	55275	obtient les paramètres pour POKE/WAIT
D7F7	55287	convers. flottant/entier
D80D	55309	effectue PEEK
D824	55332	effectue POKE
D82D	55341	effectue WAIT
D849	55369	ajoute 0.5
D850	55376	opérande de soustraction
D853	55379	effectue soustraction
D86A	55402	effectue addition
D947	55623	complémente ACC1
D97E	55683	multiplie par octet nul
D9EA	55786	effectue LOG
DA2B	55851	effectue multiplication
DA59	55897	multiplication d'un bit
DA8C	55948	mémoire vers ACC2
DAB7	55991	ajuste ACC1/ACC2
DAD4	56020	dépassement/sous-dép.
DAE2	56034	multiplie par 10
DB12	56082	effectue division
DBA2	56226	mémoire vers ACC1
DBC7	56263	ACC1 vers mémoire
DBFC	56316	ACC2 vers ACC1
DC0C	56332	ACC1 vers ACC2
DC1B	56347	arrondit ACC1
DC2B	56363	obtient le signe
DC39	56377	effectue SGN
DC58	56408	effectue ABS
DC5B	56411	compare ACC1 et mémoire
DC9B	56475	convers. flottant/entier

HEXA	*DEC*	*Rôle*
DCCC	56524	effectue INT
DCF3	56563	chaîne vers ACC
DD7E	56702	prend chiffre ASCII
DF16	57110	constantes décimales
DF3A	57146	constantes pour TI
DF71	57201	effectue SQR
DF7B	57211	effectue "puissance"
DFB4	57268	effectue "opposé"
DFED	57325	effectue EXP
E040	57408	évaluation de série (1)
E056	57430	évaluation de série (2)
E094	57492	effectue RND
E0F6	57590	points d'arrêt
E127	57639	effectue SYS
E153	57683	effectue SAVE
E162	57698	effectue VERIFY
E165	57701	effectue LOAD
E1BB	57787	effectue OPEN
E1C4	57796	effectue CLOSE
E1D1	57809	paramètres pour LOAD/SAVE
E203	57859	vérifie param. par défaut
E20B	57867	teste virgule
E216	57878	paramètres pour OPEN/CLOSE
E261	57953	effectue COS
E268	57960	effectue SIN
E2B1	58033	effectue TAN
E30B	58123	effectue ATN
E378	58232	initialisation
E387	58247	routine CHRGET (sera copiée en page 0)
E3A4	58276	initialise Basic
E429	58409	message à la mise sous tension
E44F	58447	vecteurs/copiés en $300
E45B	58459	initialise les vecteurs
E467	58471	départ à chaud
E476	58486	zone de modifications
E4A0	58528	émission série : "1"
E4A9	58537	émission série : "0"
E4B2	58546	entrée série donnée/horl
E4BC	58556	zone de modifications
E500	58624	prépare adresses 6522
E505	58629	établit limites écran
E518	58648	initialise E/S
E54C	58700	normalise écran
E55F	58719	vide écran
E581	58753	curseur à l'origine
E587	58759	établit pointeurs écran
E5BB	58811	établit val par défaut

HEXA	*DEC*	*Rôle*
E5C3	58819	prépare val par défaut dans le 6561
E5CF	58831	entrée depuis le clavier
E64F	58959	entrée depuis l'écran
E6B8	59064	teste guillemet
E6C5	59077	prépare impression écran
E6EA	59114	avance curseur
E715	59157	recule curseur
E72D	59181	recule dans la ligne précédente
E742	59202	sortie écran
E8C3	59587	va à la ligne
E8D8	59608	retour chariot
E8E8	59624	vérifie diminution de ligne
E8FA	59642	vérifie augmentation de ligne
E912	59666	établit code couleur
E921	59681	table des codes couleur
E929	59689	conversion de code
E975	59765	déroulement écran
E9EE	59886	fait place sur l'écran
EA56	59990	déplace ligne sur écran
EA6E	60014	transfert synchro couleur
EA7E	60030	établit début de ligne
EA8D	60045	vide ligne écran
EAA1	60065	imprime sur écran
EAAA	60074	range sur écran
EAB2	60082	synchro couleur caract
EABF	60095	interruption IRQ
EB1E	60190	teste clavier
EC00	60416	met en mode texte
EC46	60486	vecteurs clavier
EC5E	60510	carte clavier
ED21	60705	controle graphique/texte
ED30	60720	met en mode graphique
ED5B	60763	referme ligne écran
ED6A	60778	matrice des touches avec SHIFT
EDA3	60835	matrice des touches avec CTRL
EDE4	60900	options par défaut 6561
EDFD	60925	adr. basse ligne écran
EE14	60948	envoie "parle"
EE17	60951	envoie "écoute"
EE1C	60956	envoie caract. de commande
EE49	61001	envoi sur bus série
EEB7	61111	time-out sur bus série
EEC0	61120	envoie adresse secondaire écouteur
EEC5	61125	RAZ ATN
EECE	61134	envoie adresse secondaire parleur
EEE4	61156	envoi série différé
EEF6	61174	envoie cessation émission

HEXA	*DEC*	*Rôle*
EF04	61188	envoie cessation écoute
EF19	61209	réception sur bus série
EF84	61316	met à 1 ligne horloge
EF8D	61325	met à 0 ligne horloge
EF96	61334	délai 1ms
EFA3	61347	envoi sur RS232 (NMI)
EFEE	61422	envoie nouvel octet RS232
F016	61462	erreur ou arrêt
F027	61479	compte de bits
F036	61494	reçoit sur RS232 (NMI)
F05B	61531	prépare réception
F09D	61597	erreur parité en réception
F0A2	61602	erreur surcharge réception
F0A5	61605	erreur rupture en réception
F0A8	61608	erreur cadrage en réception
F0B9	61625	mauvais périphérique
F0BC	61628	fichier vers RS232
F0ED	61677	envoi vers tampon RS232
F116	61718	INPUT depuis tampon RS232
F14F	61775	GET depuis tampon RS232
F160	61792	vérifie bus série au repos
F174	61812	messages
F1E2	61922	imprime si mode direct
F1F5	61941	lecture ...
F205	61957	sur RS232
F20E	61966	entrée
F250	62032	GET cassette/série/RS232
F27A	62074	sortie ...
F290	62096	sur cassette
F2C7	62151	établit périph d'entrée
F309	62217	établit périph de sortie
F34A	62282	CLOSE
F3CF	62415	trouve fichier
F3DF	62431	prépare fichier
F3EF	62447	supprime tous les fichiers
F3F3	62451	rétablit périph standard
F40A	62474	ouvre fichier
F495	62613	envoie adresse secondaire
F4C7	62663	ouvre RS232
F542	62786	charge programme
F647	63047	"SEARCHING"
F659	63065	imprime nom de fichier
F66A	63082	"LOADING/VERIFYING"
F675	63093	sauve programme
F728	63272	"SAVING"
F734	63284	incrémente horloge
F760	63328	prend l'heure

HEXA	*DEC*	*Rôle*
F767	63335	met à l'heure
F770	63344	obéit à la touche STOP
F77E	63358	messages d'erreur sur fichiers
F7AF	63407	trouve en-tête bande
F7E7	63463	écrit en-tête bande
F84D	63565	obtient adresse tampon
F854	63572	établit pointeurs début/fin tampon
F867	63591	trouve en-tête spécifique
F88A	63626	incrémente pointeur bande
F894	63636	"PRESS PLAY ..."
F8AB	63659	teste état cassette
F8B7	63671	"PRESS RECORD ..."
F8C0	63680	commence lecture bande
F8E3	63715	commence écriture bande
F8F4	63732	code commun lecture/écriture
F94B	63819	vérifie arrêt bande
F95D	63837	établit le timing
F98E	63886	lit bits (IRQ)
FAAD	64173	range caractères
FBD2	64466	RAZ pointeur
FBDB	64475	prépare nouveau caractère
FBEA	64490	commute bande
FC06	64518	écrit données
FC0B	64523	écrit sur bande (IRQ)
FC95	64661	écrit en-tête (IRQ)
FCCF	64719	restaure vecteurs
FCF6	64758	établit vecteur
FD08	64776	arrête moteur
FD11	64785	teste pointeur lecture/écriture
FD1B	64795	incrémente pointeur
FD22	64802	entrée mise sous tension
FD3F	64831	teste si ROM en A000
FD52	64850	établit noyau 2
FD8D	64909	init. constantes système
FDF1	65009	vecteurs IRQ
FDF9	65617	initialise registres E/S
FE49	65097	sauve nom des données
FE50	65104	sauve détails fichiers
FE57	65111	prend l'état
FE66	65126	marque ST
FE6F	65135	marque time-out
FE73	65139	cherche haut de mémoire
FE82	65154	cherche bas de mémoire
FE91	65169	teste case mémoire
FEA9	65193	entrée interruption NMI
FED2	65234	départ à chaud RESET/STOP
FEDE	65246	séquences NMI RS232

HEXA	*DEC*	*Rôle*
FF56	65366	restaure et sort
FF5C	65372	table de temps RS232
FF72	65394	entrée IRQ principale
FF8A	65418	table de vecteurs dont :
FFC6	65478	établit canal d'entrée
FFC9	65481	établit canal de sortie
FFCC	65484	restaure canaux d'E/S standard
FFCF	65487	entrée (INPUT)
FFD2	65490	sortie (PRINT)
FFE1	65505	teste touche STOP
FFE4	65508	saisie (GET)
FFFA	65730	vecteurs hardware.

ADRESSES DES ROUTINES IEEE

Adresse	*Vecteur*	*Rôle*
E4A0 58528		Met à 1 la ligne de donnée
E4A9 58537		Met à 0 la ligne de donnée
E4B2 58546		Décale l'état de la ligne horloge vers la retenue
EE14 60948	FFB4 65460	Commande "parle"
EE17 60951	FFB1 65457	Commande "écoute"
EE40 60992		Envoi d'un octet
EE6F 61039		Prépare émission
EEC0 61120	FF93 65427	Envoie adresse secondaire écouteur
EEC5 61125		Relâche ATN
EECE 61134	FF96 65430	Envoie adresse secondaire parleur
EED3 61139		ATN parleur
EEE4 61156	FFA8 65448	Envoie octet du tampon
EEF6 61174	FFAB 65451	Commande "cessation d'émission"
EF04 61188	FFAE 65454	Commande "cessation d'écoute"
EF19 61209	FFA5 65445	Obtient un octet du bus
EF84 61316		Met à 1 la ligne horloge

Adresse	*Vecteur*	*Rôle*
EF8D 61325		Met à 0 la ligne horloge
EF96 61334		Délai 1 ms

Variables : (adresse hexa. déc.)

90	144	Indicateur d'état
94	148	Indicateur de tampon
95	149	Tampon caractère en sortie
98	152	Pointeur dans la table des fichiers = nombre de fichiers ouverts
9A	154	Numéro de périphérique standard d'entrée
9B	155	Numéro de périphérique standard de sortie
A3	163	Indicateur d'EOI
B7	183	Longueur du nom de fichier courant
B8	184	Numéro logique courant
B9	185	Adresse secondaire courante
BA	186	Numéro de périphérique courant
BB	187	Pointeur vers nom de fichier

EFA3	Entrée NMI
EFBF	Calcule parité
EFEE	Départ émission d'un octet
F036	Routine NMI d'assemblage des octets
F05B	Prépare réception
F08B	Vérifie la parité
F0BC	Sortie sur fichier RS232
F0C4	Test DSR et RTS
F0D4	Attend CTS à 0
F0D9	Met RTS à 1
F0E1	Attend CTS à 1
F0ED	Gestion du tampon sortie
F116	Entrée sur fichier RS232
F12B	Attend fin de la sortie active
F130	Met RTS à 0
F14F	Gestion du tampon d'entrée

Variables (hexa - déc.)

0293	659	Registre de contrôle du pseudo 6551
0294	660	Registre de commande du pseudo 6551
0297	663	Registre d'état
AA	170	Tampon caractère où on assemble les bits reçus
B6	182	Tampon caractère où on sérialise les octets à envoyer
F7-F8	247-248	Pointeur vers le tampon réception
F9-FA	249-250	Pointeur vers le tampon émission
29B	667	Index vers fin du tampon réception
29C	668	Index vers début du tampon réception
29D	669	Index vers début du tampon émission
29E	670	Index vers fin du tampon émission

WRT	($FFD2) sortie du caractère dont le code est en A.
STROUT	($CB1E) sortie de la chaîne pointée par Y,A.
CRLF	($CAD7) : fait un retour-chariot.
LF	($E519) : va à la ligne.
CLR	($E55F) : vide l'écran.
SCROLL	($E975) déroule l'écran d'une ligne vers le bas. (Pour dérouler vers le haut, c'est $E9EE.
GETIN	($FFE4) saisit un caractère au vol (rend son code ASCII dans l'accumulateur). Retourne 0 s'il n'y a pas de caractère.
CHRIN	($FFCF) attend qu'on frappe un caractère : lorsqu'on l'a tapé, renvoie son code en A.
INPUT	($C560) simule l'INPUT du Basic : on accepte une chaîne de caractères terminée par RETURN et on l'envoie dans le tampon d'entrée de Basic ($200-258).

ROUTINES ARITHMETIQUES

Les routines arithmétiques utilisent deux accumulateurs ACC1 $61-66) et ACC2 ($69-6E) avec le format voulu pour contenir les nombres réels.

MACC1 ($DAD3) transfère dans ACC1 ($61...) un nombre au format flottant pointé par A(bas) Y(haut). On effectue en même temps la conversion du format flottant ordinaire en format accumulateur.

MACC2 ($DA8C) joue le même rôle mais vers ACC2.

ACC1M ($D8D4) transfère ACC1 vers le groupe de 5 octets pointé par X(b) Y(h), avec conversion du format accumulateur en format flottant ordinaire.

Il n'y a pas de ACC2M car le résultat d'une opération se retrouve toujours en ACC1.

ACC21 ($DBFC) transfère ACC2 dans ACC1.

ACC12 ($DCØC) transfère ACC1 dans ACC2.

ZERO ($D8F7) met ACC1 à zéro.

OPPOS ($D947) transforme ACC1 en son opposé.

ARRON ($DC1B) arrondit ACC1.

NORMAL ($D8D7) normalise ACC1 (remet sous la forme 0)

Nom	*Adresse*	*Opération*
ADD	D86A	ACC1←ACC1+ACC2
ADDM	D867	ACC1←ACC1+(Y,A)
SUB	D853	ACC1←ACC1-ACC2
SUBM	D850	ACC1←ACC1-(Y,A)
MULT	DA30	ACC1←ACC1*ACC2
MULTM	DA28	ACC1←ACC1*(Y,A)
DIV	DB12	ACC1←ACC2/ACC1
DIVM	DBOF	ACC1←(Y,A)/ACC1
FPEXP	DF7B	ACC1← ACC2 ↑ACC1
CMPO	DC2B	A= FF,00,01 si ACC1 < = > 0

Nom	*Adresse*	*Opération*
CMPM	DC5B	A= FF,00,01 si ACC1 < = > M

Les fonctions de bibliothèque arithmétique effectuent l'opération ACC1←f(ACC1). Voici les adresses de la bibliothèque Basic.

Fonction	*adresse*
ABS	DC58
ATN	E30B
COS	E261
EXP	DFED
INT	DCCC
LOG	D9EA
SIN	E268
SQR	DF71
TAN	E2B1

Routines de conversion

INTFLP ($D391) convertit en flottant dans ACC1 le nombre entier dont la partie haute est dans A et la partie basse dans Y.

FLPINT ($D1BF) convertit en entier le nombre flottant qui est dans ACC1. Le résultat est placé dans ACC1 : partie haute en $64, partie basse en $65.

FLPASC ($DDDD) convertit en ASCII le nombre contenu dans ACC1. Le résultat sera à partir de $0100 et terminé par un octet 00, c'est-à-dire tout prêt à être imprimé par STROUT si l'on charge A,Y pour pointer vers 0100 :
LDA = $00
LDY = $01

Conversion d'une chaîne en nombre

Entier : faire pointer $7A-7B vers la chaîne (terminée par un blanc) puis :
JSR CHRGET ; $0073
JSR ASCINT ; $96B

Le résultat sera en $14 (bas), $15 (haut).

Réel : faire pointer $7A-7B vers la chaîne (terminé par une virgule), puis :

JSR EXPR ; $CD9E.

EXPR est la routine d'évaluation d'expressions arithmétiques que nous avons déjà utilisée. Attention, en utilisation avec Basic, le pointeur ($7A-7B) doit être sauvegardé et restauré !

2B,2C	43,44	Début du texte Basic.
2D,2E	45,46	Fin de Basic, début des variables.
2F,30	47,48	Fin des variables, début des tableaux.
31,32	49,50	Fin des tableaux, début de l'espace libre.
33,34	51,52	Fin de l'espace libre, début des chaînes.
35,36	53,54	Fin des chaînes.
37,38	55,56	Fin de l'espace-mémoire.
7A,7B	122,123	Pointeur dans le texte-source pour CHRGET.

ADRESSES DES VARIABLES STRATEGIQUES DU SYSTEME DU VIC

Hexa	*Dec.*	*Description*
0000-0002	0-2	Saut pour USR (adresse en 1-2)
0003-0004	3-4	Vecteur pour conversion réel-entier
0005-0006	5-6	Vecteur pour conversion entier-réel
0007	7	Caractère recherché
0008	8	Indicateur mode guillemets
0009	9	Sauvegarde colonne de tabulation
000A	10	0=LOAD 1=VERIFY
000B	11	Pointeur dans tampon d'entrée/numéro d'indice
000C	12	Indicateur de DIM par défaut
000D	13	Type : $FF = chaîne 00 = numérique
000E	14	Type : $80 = entier 00 = réel
000F	15	Indicateur mémoire, mode guillemets en LIST, balayage DATA
0010	16	Indicateur indice / FNx
0011	17	0=INPUT, $40=GET, $98=READ
0012	18	Signe pour ATN/indicateur de comparaisons
0013	19	Indicateur actuel d'interrogation d'entrée
0014-0015	20-21	Valeur entière
0016	22	Rangement temporaire du pointeur de pile
0017-0018	23-24	Dernier vecteur de chaîne temporaire
0019-0021	25-33	Pile pour chaînes temporaires
0022-0025	34-37	Zone de pointeurs utilitaires
0026-002A	38-42	Zone pour le produit dans une multiplication
002B-002C	43-44	Pointeur : début de Basic
002D-002E	45-46	Pointeur : début des variables
002F-0030	47-48	Pointeur : début des tableaux
0031-0032	49-50	Pointeur : fin des tableaux
0033-0034	51-52	Pointeur : zone des chaînes
0035-0036	53-54	Pointeur utilitaire chaînes
0037-0038	55-56	Pointeur : fin de mémoire
0039-003A	57-58	Numéro de ligne Basic courante
003B-003C	59-60	Numéro de ligne Basic précédente
003D-003E	61-62	Pointeur instruction Basic pour CONT
003F-0040	63-64	Numéro de ligne DATA courante
0041-0042	65-66	Adresse courante de DATA
0043-0044	67-68	Vecteur pour INPUT
0045-0046	69-70	Nom de la variable courante
0047-0048	71-72	Adresse de la variable courante
0049-004A	73-74	Pointeur vers variable pour FOR...NEXT
004B-004C	75-76	Sauvegarde de Y ou d'opérateur ou du pointeur Basic
004D	77	Accumulateur des symboles de comparaison
004E-0053	78-83	Zone de travail diverse
0054-0056	84-86	Vecteur de saut pour fonctions
0057-0060	87-96	Zone de travail numérique diverse

Hexa	*Dec.*	*Description*
0061	97	Acc. numéro 1 : exposant
0062-0065	98-101	Acc. numéro 1 : mantisse
0066	102	Acc. numéro 1 : signe
0067	103	Pointeur vers constantes pour évaluer une série
0068	104	Débordement de Acc. numéro 1
0069-006E	105-110	Acc. numéro 2 : exposant, mantisse, signe
006F	111	Signe de la comparaison Acc.1 Acc.2
0070	112	Arrondi de l'Acc. numéro 1
0071-0072	113-114	Longueur tampon-cassette/pointeur série
0073-008A	115-138	Sous-programme CHRGET (obtient un caractère Basic)
007A-007B	122-123	Pointeur Basic (dans CHRGET)
008B-008F	139-143	Racine pour RND
0090	144	Mot d'état ST
0091	145	Copie du PIA clavier : indicateurs de STOP et RVS
0092	146	Constante de vitesse pour cassette
0093	147	0 = LOAD ; 1 = VERIFY
0094	148	Sortie série : indicateur de caractère différé
0095	149	Caractère série différé
0096	150	EOT reçue sur bande
0097	151	Sauvegarde de registre
0098	152	Nombre de fichiers ouverts
0099	153	Périphérique standard d'entrée (normalement 0)
009A	154	Périphérique de sortie standard déterminé par CMD (normalement 3)
009B	155	Parité du caractère sur cassette
009C	156	Indicateur de caractère reçu
009D	157	Contrôle de sortie : directe =$80,RUN=0
009E	158	Témoin d'erreur sur cassette passe 1/ tampon caractère
009F	159	Témoin d'erreur corrigée sur cassette (passe 2)
00A0-00A2	160-162	Horloge au 60e de seconde
00A3	163	Compte de bits série/indicateur d'EOI
00A4	164	Compteur de cycles
00A5	165	Décompteur écriture cassette/compteur de bits
00A6	166	Pointeur dans tampon cassette
00A7	167	Compteur en-tête écriture cassette/passe lecture/bit entrée
00A8	168	nouvel octet écriture cassette/ erreur lecture/ compte bits entrées

ADRESSES DES VARIABLES STRATEGIQUES DU SYSTEME DU VIC

Hexa	*Dec.*	*Description*
00A9	169	Ecrit bit départ/erreur lecture bit/ bit de départ
00AA	170	Balayage cassette/assemblage octet
00AB	171	Longueur en-tête écriture/checksum lecture/parité
00AC-00AD	172-173	Pointeur : tampon cassette, déroulement écran
00AE-00AF	174-175	Adresse fin cassette/fin du programme
00B0-00B1	176-177	Constantes de vitesse pour cassette
00B2-00B3	178-179	Pointeur : début du tampon cassette
00B4	180	1 = temporisateur cassette activé/ compte de bits
00B5	181	EOT cassette/prochain bit à envoyer en RS232
00B6	182	Erreur caractère en lecture/tampon octet à sortir
00B7	183	Nombre de caractères du nom de fichier
00B8	184	Fichier logique courant
00B9	185	Adresse secondaire courante
00BA	186	Périphérique courant
00BB-00BC	187-188	Pointeur vers nom de fichier
00BD	189	Mot à décaler en sortie/caractère entré
00BE	190	Nombre de blocs restant à lire ou écrire
00BF	191	Tampon mot série
00C0	192	Commande moteur cassette
00C1-00C2	193-194	Adresse départ d'E/S
00C3-00C4	195-196	Pointeur routine de préparation du noyau
00C5	197	Dernière touche appuyée
00C6	198	Nombre de caractères dans le tampon clavier
00C7	199	Indicateur de contraste inversé
00C8	200	Pointeur vers fin de ligne en entrée
00C9-00CA	201-202	Témoin curseur en entrée (ligne/colonne)
00CB	203	Touche enfoncée : 64 si aucune
00CC	204	0 = le curseur clignote
00CD	205	Décompteur pour vitesse de clignotement curseur
00CE	206	Caractère sous le curseur
00CF	207	Phase de clignotement curseur
00D0	208	Entrée depuis l'écran ou le clavier
00D1-00D2	209-210	Pointeur vers ligne écran
00D3	211	Position du curseur dans la ligne
00D4	212	0 = curseur direct ; sinon programmé
00D5	213	Longueur ligne courante sur écran
00D6	214	Ligne où se trouve le curseur
00D7	215	Tampon/checksum/dernière touche
00D8	216	Nombre d'insertions en attente

Hexa	*Dec.*	*Description*
00D9-00F0	217-240	Table des liens des lignes-écran
00F1	241	Lien bidon
00F2	242	Marqueur ligne écran
00F3-00F4	243-244	Pointeur mémoire couleur écran
00F5-00F6	245-246	Pointeur clavier
00F7-00F8	247-248	Pointeur vers réception RS232
00F9-00FA	249-250	Pointeur vers émission RS232
0100-01FF	256-511	Pile processeur, dont sont utilisés temporairement :
00FF-010A	255-266	Zone de travail pour la conversion réel ASCII
0100-013E	256-318	Mémorisation erreurs cassette
0200-0258	512-600	Tampon d'entrée du Basic
0259-0262	601-610	Table des fichiers logiques
0263-026C	611-620	Table des numéros de périphériques
026D-0276	621-630	Table des adresses secondaires
0277-0280	631-640	Tampon clavier
0281-0282	641-642	Plus petite adresse de RAM Basic
0283-0284	643-644	Plus grande adresse de RAM Basic
0285	645	Indicateur de time-out sur le bus série
0286	646	Code couleur courant
0287	647	Couleur sous le curseur
0288	648	Page de la mémoire d'écran
0289	649	Taille maximum du tampon d'écran (doit être < 10)
028A	650	Répétition automatique des touches : 0 = touches curseur ; 127 = aucune touche 128 = toutes les touches
028B	651	Compteur pour la vitesse de répétition
028C	652	Compteur du délai pour la répétition
028D	653	Indicateur de SHIFT, 'C=', CTRL: 0=rien, 1=SHIFT, 2='C=', 4=CTRL
028E	654	Dernier motif de SHIFT
028-0290	655-656	Pointeur vers la routine d'établissement de la table de transcodage clavier
0291	657	Mode des touches (Katakana ou non ; aucune utilité dans les versions européennes)
0292	658	0 = déroulement écran autorisé
0293	659	Contrôle du boîtier VIC
0294	660	Commande du boîtier VIC
0295-0296	661-662	Temps d'un bit pour le RS232
0297	663	Etat du RS232
0298	664	Nombre de bits à envoyer
0299-029A	665-666	Code vitesse du RS232
029B	667	Pointeur réception RS232
029C	668	Pointeur entrée RS232

ADRESSES DES VARIABLES STRATEGIQUES DU SYSTEME DU VIC

Hexa	*Dec.*	*Description*
029D	669	Pointeur émission RS232
029E	670	Pointeur sortie RS232
029F-02A0	671-672	Sauvegarde IRQ durant E/S cassette
0300-0301	768-769	Pointeur message d'erreur
0302-0303	770-771	Pointeur vers reset tiède de Basic
0304-0305	772-773	Pointeur vers routine de codage des mots-clés Basic
0306-0307	774-775	Pointeur vers routine d'impression des codes des mots-clés
0308-0309	776-777	Pointeur vers départ nouvelle ligne Basic
030A-030B	778-779	Pointeur vers routine d'obtention d'un élément arithmétique
030C-030F	780-783	Zone à partir de laquelle sont chargés A,X,Y et P lors d'un SYS.
0314-0315	788-789	Vecteur d'interruption hardware ($EABF)
0316-0317	790-791	Vecteur de BRK (FED2)
0318-0319	792-793	Vecteur de NMI (FEAD)
031A-031B	794-795	Vecteur pour OPEN (F40A)
031C-031D	796-797	Vecteur pour CLOSE (F34A)
031E-031F	798-799	Vecteur pour établir l'entrée standard (F2C7)
0320-0321	800-801	Vecteur pour établir la sortie standard (F309)
0322-0323	802-803	Vecteur pour restaurer les E/S standard (F3F3)
0324-0325	804-805	Vecteur pour INPUT (F20E)
0326-0327	806-807	Vecteur pour sortie (F27A)
0328-0329	808-809	Vecteur pour tester STOP (F770)
032A-032B	810-811	Vecteur pour GET (F1F5)
032C-032D	812-813	Vecteur pour abandonner les E/S (F3EF)
032E-032F	814-815	Vecteur pour USR (FED2)
0330-0331	816-817	Pointeur pour LOAD
0332-0333	818-819	Pointeur pour SAVE
033C-03FB	828-1019	Tampon cassette
0400-0FFF	1024-4095	Zone d'extension RAM 3K
1000-1FFF	4096-8191	Mémoire RAM d'un VIC de base
2000-7FFF	8192-32767	Zone d'extension mémoire

Vecteur RAM		*Vecteur ROM*		*Adresse*		*Rôle*
hexa	*dec*	*hexa*	*dec*	*hexa*	*dec*	
0300	768			C43A	50234	message d'erreur
0302	770			C483	50307	départ à chaud de Basic
0304	772			C57C	50556	code les mots-clés Basic
0306	774			C71A	50970	imprime les mots-clés
0308	776			C7E4	51172	départ nouvelle instruction Basic
030A	778			CE86	52870	prend un élément arithmétique
0314	788			EABF	60095	interruption IRQ
0316	790			FED2	65234	interruption BRK
0318	792			FEAD	65197	interruption NMI
031A	794	FFC0	65472	F40A	62474	OPEN
031C	796	FFC3	65475	F34A	62282	CLOSE
031E	798	FFC6	65478	F2C7	62151	établit entrée standard
0320	800	FFC9	65481	F309	62217	établit sortie standard
0322	802	FFCC	65484	F3F3	62451	restaure entrées/sorties standard
0324	804	FFCF	65487	F20E	61966	INPUT
0326	806	FFD2	65490	F27A	62074	routine de sortie
0328	808	FFE1	65505	F770	63344	teste STOP
032A	810	FFE4	65508	F1F5	61941	GET au clavier
032C	812	FFE7	65511	F3EF	62447	abandonne les entrées/sorties
032E	814			FED2	65234	vecteur utilisateur
0330	816	FFD5	65493	F549	62793	LOAD
0332	818	FFD8	65496	F685	63109	SAVE
		FF8A	65418	FD52	64850	restaure vecteurs standard
		FF8D	65421	FD57	64855	lit et établit les vecteurs
		FF90	65424	FE66	65126	contrôle les messages
		FF93	65427	EEC0	61120	envoie adresse secondaire écouteur
		FF96	65430	EECE	61134	envoie adresse secondaire parleur

VECTEURS

Vecteur RAM		*Vecteur ROM*		*Adresse*		*Rôle*
hexa	*dec*	*hexa*	*dec*	*hexa*	*dec*	
		FF99	65433	FE73	65139	lit/établit haut de mémoire
		FF9C	65436	FE82	65154	lit/établit bas de mémoire
		FF9F	65439	EB1E	60190	balayage clavier
		FFA2	65443	FE6F	65135	établit time-out sur IEEE
		FFA5	65445	EF19	61209	reçoit un octet de l'IEEE
		FFA8	65448	EEE4	61156	sort un octet sur l'IEEE
		FFAB	65451	EEF6	61174	commande cessation d'émission
		FFAE	65454	EF04	61188	commande cessation d'écoute
		FFB1	65457	EE17	60951	commande "écoute"
		FFB4	65460	EE14	60948	commande "parle"
		FFB7	65463	FE57	65111	lit le mot d'état ST
		FFBA	65466	FE50	65104	établit n°périph,n°logique, ad.sec.
		FFBD	65469	FE49	65097	établit nom de fichier
		FFDB	65499	F767	63335	met l'horloge à l'heure
		FFDE	65502	F760	63328	lit l'horloge temps réel
		FFEA	65514	F734	63284	incrémente l'horloge
		FFED	65517	E505	58629	renvoie l'organisation de l'écran
		FFF0	65520	E50A	58634	lit/impose position du curseur
		FFF3	65523	E500	58624	renvoie l'adresse de base des boîtiers d'E/S
		FFFA	65530	FEA9	65193	NMI
		FFFC	65532	FD22	64802	RESET
		FFFE	65534	FF72	65394	IRQ

CBM	VIC
0 . 2	0 . 2
3 . 4	7 . 8
5 : 13	11 : 19
17 : 140	20 : 143
141 . 143	160 . 162
144 : 149	788 : 793
150	144
151	203
152	653
155 . 157	145 . 147
158 159	198 199
160	148
161	200
163 164	201 202
165	149
166	197
167 . 170	204 . 207
172	208
173 : 181	151 : 159

CBM	VIC
183	163
185 . 190	164 . 168
193 195	169 171
196 198	209 211
199 . 204	172 . 177
205	212
206 . 212	180 . 186
213	213
214 215	178 179
216 217	214 215
218 219	187 188
220	216
221 223	189 191
224 . 248	217 . 241
249	192
251 252	193 194
256 . 511	256 . 511
512 . 591	512 . 600
593 . 632	601 . 640

CONVERSION CBM/VIC

CBM	*VIC*
634 • 825	828 • 1019

CBM 4000G	*CBM 8000*	*VIC*
227	1003	649
228	1000	650
229	1001	651
	1002	652

N.B. Les correspondances entre adresses concernant la gestion des cassettes sont données sous réserves.

VIC	CBM
0 . 2	0 . 2
7 8	3 4
11 . 19	9 . 13
20 : 143	17 : 140
144	150
145 . 147	155 . 157
148	160
149	165
150	207
151 : 159	173 182
160 . 162	141 . 143
163	183
164 . 166	185 . 187
167 168	189 190
169 . 171	193 . 195
172 . 177	199 . 204
178 179	214 215
180 . 186	206 . 212

VIC	CBM
187 188	218 219
189 . 191	221 . 223
192	249
193 . 194	251 . 252
197	166
198 199	158 159
200	161
201 202	163 164
203	151
204 . 207	169 . 170
208	172
209 211	196 198
212	205
213	213
214 215	216 217
216	220
217 : 241	224 : 248
256 : 511	256 : 511
512 : 600	512 : 592
601 : 640	593 : 632

CONVERSION VIC/CBM

VIC	*CBM*
645	1020 (4.8000)
649	227 (8000) 1003 (4000 G)
650	228 (8000) 1006 (4000 G)
651	229 (8000) 1001 (4000 G)
652	1002 (4000 G)
653	152
788 : 793	144 : 149
828 : 1019	634 : 825

N.B. les correspondances des adresses concernant la gestion des cassettes sont données sous réserves.

ADRESSES SECTEURS DISQUES

Unité	*1540*	
Catalogue et BAM	Piste 18 sect. 0 et 1	
Pistes	secteurs	nb
1-17	0-20	21
18-24	0-18	19
25-30	0-17	18
31-35	0-16	17

LES "COMMENT...?"

1 - Positionner le curseur en ligne L, colonne C

```
POKE 781,L-1:POKE 782,C-1:POKE 783,Ø:SYS 65520
```

2 - Simuler INPUT

Assembleur	*Basic*	*La chaîne lue va en :*
JSR $C560	SYS 50528	512-599

3 - Empêcher le listing d'un programme

```
POKE 1Ø25,Ø (si ext.3K)  POKE 46Ø9,Ø (si ext. >8k)
POKE 4Ø97,Ø (si VIC de base)
```

4 - Sauver une table sur périphérique

```
                          avec :
                          NOM$ : nom du fichier
POKE 43,D2                P    : numéro de périphérique
POKE 44,D1                D1   : adresse de début haute
POKE 45,F2                D2   : adresse de début basse
POKE 46,F1                F1   : adresse de fin haute
SAVE NOM$,P               F2   : adresse de fin basse
```

5 - Lire et assigner les touches de fonction

Les 4 touches situées à droite du clavier du VIC se lisent exactement comme les autres par PEEK (203) ou par leur code ASCII obtenu par GET A$:PRINT ASC(A$).

Les valeurs sont :

Touche	*PEEK (203)*	*PEEK (653)*	*ASC (A$)*
F1	39	0	133
F2	39	1	137
F3	47	0	134

Touche	*PEEK (203)*	*PEEK (653)*	*ASC (A$)*
F4	47	1	138
F5	55	0	135
F6	55	1	139
F7	63	0	136
F8	63	1	140

Pour assigner une fonction à la touche, il suffit alors
de IF PEEK (203) = ... GOTO ...
ou IF ASC (A$) = ... GOTO ...

Si l'on veut que l'appui de la touche soit équivalent à l'appui d'une certaine chaîne de caractères, c'est simple avec PROGRAMMER'S AID ou SUPER EXPANDER : il y a la commande KEY.

En l'absence de ces cartouches, il faut détourner la routine d'interruption et, dans votre routine, déceler la touche de fonction et forcer la chaîne souhaitée dans le tampon clavier.

Exemple (entré sous moniteur)

```
$1500   LDA   $CB       ; adresse 203
$1502   CMP   #$27      ; 39 = touche F1
$1504   BEQ   $1509     ; si oui va traiter la touche
$1506   JMP   $6346     ; sinon retourne à l'interruption normale
$1509   LDA   #$41      ; lettre A
$150B   STA   $0277     ; met AA dans le tampon clavier
$15DE   STA   $0278
$1511   LDA   #$02
$1513   STA   $C6       ; il y a 2 caractères dans le tampon
$1515   JMP   $6346     ; retour à l'interruption normale
```

On active cette fonction qui consiste à ce que F1 soit équivalente à "AA" en mettant 00 en $314 et $15 en $315.

6 - Inhiber la touche STOP

POKE 788, PEEK (788)+3 si on est sûr qu'elle n'est pas déjà inhibée.

Pour la rétablir (si on est sûr qu'elle est inhibée) :
POKE 788, PEEK (788)-3

7 - Attendre qu'on appuie sur SHIFT

WAIT SH,1 SH = 653
attendre qu'on relâche la touche : WAIT SH,1,1

8 - Attendre un caractère quelconque au clavier

```
POKE  198,Ø
WAIT  198,1
```

9 - Faire une pause dans un listing

Il peut être commode qu'un listing soit interrompu lorsqu'on appuie sur SHIFT et reprenne lorsqu'on lâche.

Là encore on détourne la routine d'interruption.

```
$1A00     SEI                ; routine d'activation : lancement par
                               SYS 6656
$1A01     LDA   $0314        ; sauve ancien vecteur
$1A04     STA   $029F
$1A07     LDA   $0315
$1A0A     STA   $02A0
$1A0D     LDA   #$19         ; prépare nouveau vecteur
$1A0F     STA   $0314
$1A12     LDA   #$1A
$1A14     STA   $0315
$1A16     LDA   $028D
$1A17     CLI
$1A18     RTS
$1A19     JSR   $FF9F        ; entrée interruption
$1A1F     BNE   $1A19        ; si SHIFT enfoncée, boucle.
$1A21     JMP   ($029F)      ; saute à l'interruption normale.
```

Ceci cesse de fonctionner si l'on utilise les cassettes.

10 - Indenter un programme Basic

On utilise : ou n'importe quel caractère graphique.

11 - GOTO calculé ou SYS avec arguments

On écrit SYS adresse, expression

avec en adresse
```
JSR VIRG ; vérifie virgule
JSR EXPR ; calcule expression
```

utilise argument dans ACC1 pour GOTO
```
JSR INTE ; convertit en entier
JSR GOTO ; effectue le GOTO
```

Les adresses sont :

	VIC		*VIC*
VIRG	CEFD	INTE	D7F7
EXPR	CD8A	GOTO	C8A3

12 - Imprimer avec D décimales

```
DEF FNF(X)=INT(X*10↑D)/10↑D
```

et au lieu d'imprimer X, on imprime FNF(X)

13 - Justifier à droite dans une zone de C caractères

C$ ="◄——————►" C blancs

?RIGHT$(C$+A$,C) (peut être combiné avec 12 si A$=STR$(FNF(X)))

14 - Charger un fichier/programme sans translation

On fait LOAD "NOM",périph,mode
avec périph = 1 ou 8 (1 par défaut)
et mode = 1 (par défaut, 0 = translation)

15 - Définir une fonction lors de l'exécution

(par exemple pour un programme de calcul d'intégrale) :

```
1  INPUT"QUELLE FONCTION VOULEZ-VOUS INTEGRER",F$
2  PRINT"CLR";"1Ø DEF FNF(X)="F$
3  CL=631 : PT = 198
4  POKE CL,ASC("Home"):POKE CL+1,13:POKE CL+2,ASC("G")
5  POKE CL+3,ASC("O"):POKE CL+4,ASC("1"):POKE CL+5,ASC("Ø")
6  POKE CL+6,13:POKE PT,7:END
1Ø DEF FNF(X)=EXP(X)
2Ø suite
```

16 - Connecter la télé

Les branchements suivants sont possibles :

```
VIC... interface N/B ... prise antenne télé SECAM (image N/B)
VIC... interface Secam . prise antenne télé SECAM (image couleurs)
                         /péritel
VIC... cordon        ... prise péritel télé SECAM (image N/B)
VIC... cordon        ... prise péritel télé PAL   (image couleurs)
VIC... modulateur    ... prise antenne  télé PAL   (image couleurs)
```

Le schéma du cordon PERITEL est :

VIC connecteur vidéo		*TELE prise Péritel*	
1	+5V	8	(éventuellement)
2	masse _____	18	
3	son _____	6	
4	vidéo basse		
5	vidéo haute_____	20	

Si le branchement ci-dessus ne fonctionne pas, il faut faire la liaison VIC-1 .... péritel 8.

Si cela ne fonctionne pas, il faut plus que 5V. La Commode n°4 page 52 donne le moyen d'amener du 9V. à la broche 1.

17 - Connaître l'adresse d'une variable

Mettre code ASCII 1ère lettre du nom en $45
2e lettre en $46

Le code des caractères est modifié : on ajoute $80 au 2e code si c'est une chaîne, et aux deux codes si c'est une variable entière.

Puis JSR $D0E5 et l'adresse est en :
$47,48

Méthode purement Basic

Si A est la variable concernée :

```
100  X=A
110  A=PEEK(M)+256*PEEK(N)-2
120  PRINT A
130  A=X
```

avec M=71, N=72

18 - Lister sur imprimante

```
OPEN 4,4:CMD:LIST
```

puis :

```
PRINT#4:CLOSE4
```

19 - Passer du mode graphique à minuscules

SHIFT 'C=' ou
PRINT CHR$(142)/PRINT CHR$(14)

20 - Augmenter la largeur de l'affichage

On peut augmenter (légèrement) la largeur de l'affichage (25 col. par ligne) en supprimant les marges et en modifiant des registres du 6561 :

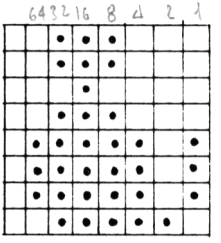
```
POKE 36864,9:POKE36866,153:POKE36867,40:POKE36865,42
```

21 - Définir des caractères programmables sur l'écran

POKE 36869,254 fixe le générateur de caractères en RAM à G=$1800=6144 ce qui permet 192 caractères programmables.

Il faut le protéger par POKE 52,24:POKE 56,24:CLR

Ensuite, le caractère de code écran Z est défini dans une matrice 8x8 entre les adresses G+8*Z et G+8*Z+7.



Par exemple, pour le petit chat ci-contre, on aurait :

```
100 DATA 56,56,16,56,125,125,125,62
110 DOR I=0 TO 7
120 READ A:POKE G+8*Z+I,A:NEXT
```

Et on obtient le petit chat par PRINT CHR$(Z+64) ou POKE écran,Z.

22 - Vider le tampon clavier

```
POKE 198,0
```

23 - Avoir le curseur pendant un GET

```
POKE 204,0
```

24 - Attendre STOP au clavier

```
WAIT 37153,1,1
```

Au lieu de 1, on peut avoir 2 (SHIFT gauche) 4(X),8(V),16(N),32(,),64(/) ou 128 (curseur bas)

25 - Effacer écran (vide la mémoire d'écran)

```
SYS 58719 ($E55F)
```

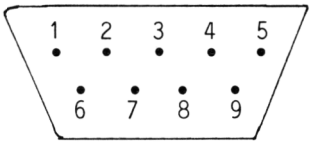
26 - Scroll (défilement de l'écran) vers le haut

```
SYS 59765 ($E975)
```

27 - Lire les joysticks

Il s'agit ici d'un manche à balai formé de 4 interrupteurs

N(JOY0,br1 du connecteur de jeux)
O(JOY2,br3) E (JOY3,br4)
S (JOY1,br2)



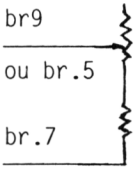
et d'un bouton "feu", relié à la br.6 du connecteur.

Les connexions sont JOY0-PA2;JOY1-PA-3;JOY2-PA4;JOY3-PB7 et FEU-PA5, avec adresse PA=37151; adresse PB=37152.

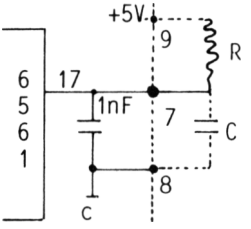
Direction	*Interrupteurs fermés*	*Test (interrupteur fermé = broche de VIA à 0)*
N	JOY0	IF PEEK(PA)AND 4 ...
N-O	JOY0-JOY2	IF PEEK(PA)AND 20 ...
O	JOY2	IF PEEK(PA)AND 16 ...
S-O	JOY1-JOY2	IF PEEK(PA)AND 24 ...
S	JOY1	IF PEEK(PA)AND 8
S-E	JOY1-JOY3	IF (PEEK(PA)AND8)+(PEEK(PB)AND 32)...
E	JOY3	IF PEEK(PB)AND 32 ...
N-E	JOY0-JOY3	IF (PEEK(PA)AND 4)+(PEEK(PB) AND 32).
"feu"	crayon lumineux	IF PEEK(PA)AND 32 ...

28 - Mesurer une résistance ou une capacité

Si on relie une résistance entre la broche POTX(9) ou POTY(5) et la broche +5V(7) du connecteur de jeux, on obtient en 36872 (X) ou 36873(Y) une valeur sur un octet (donc comprise entre 0 et 255, proportionnelle à la résistance. Il faut étalonner pour chaque VIC, mais, en gros il faut utiliser un potentiomètre de valeur maximum dans les 200-250 kohms. La valeur 0 est nuisible. Pour l'éviter on conseille le montage ci-contre, avec une résistance de garde RG de 3 kohms. La valeur obtenue par PEEK (36872) ou PEEK (36873) est, en gros, le nombre de kilohms de la résistance.



Le montage réalisé dans le VIC est le suivant (on prend le cas de X dans la figure ci-contre). On constitue un circuit RC avec le condensateur de 1nF déjà présent entre la masse et la broche POT... du 6561, et le boîtier VIC 6561 ne fait rien d'autre que mesurer la constante RC qui se retrouve dans le registre d'adresse 36872 ou 36873.



connecteur de jeux

Si l'on met un condensateur C à l'extérieur du VIC entre les broches 7 et 8 du connecteur H, on pourra mesurer sa capacité ou mesurer une résistance R dans une autre gamme que les 200 k. Selon la résistance R, on aura différentes gammes de capacités mesurables.

Dans tous les cas, un étalonnage préalable est à faire.

29 - Protéger le haut de la mémoire

On fait POKE en 51,52 et 55,56.

30 - Identifier la touche enfoncée

PEEK(TC) est la localisation de la touche sur le clavier. (TC=203).

PEEK(TC)=64 si aucune touche n'est enfoncée.

Pour avoir le code ASCII correspondant, faire :

PEEK (PEEK(TC)+TA) avec TA=60510

31 - Vérifier le fonctionnement des cassettes

Après un LOAD, PEEK(182) doit avoir une faible valeur.

32 - Relancer un programme

On peut utiliser CONT (impossible si l'on a modifié le programme)

ou GOTO numéro

ou RUN numéro (ce dernier démolit les variables).

33 - Eviter des ennuis de INPUT

1 - Lorsque la phrase d'interrogation d'un INPUT fait plus d'une ligne (donc, dépasse 22 caractères), le VIC la prend dans la réponse.

Pour l'éviter, ou bien terminer la phrase par un curseur bas ou bien commencer la réponse par un curseur haut.

2 - Pour sortir du programme lors d'un INPUT, Stop est inhibé et Return tout court donne à la variable sa valeur antérieure. Faire Stop Restore.

34 - Prévoir la taille d'un programme

En gros, un programme occupe autant d'octets qu'il renferme de caractères, puisqu'il est stocké tel quel, comme chaîne de caractères, sauf les mots-clés qui sont remplacés par un code en 1 octet.

En ce qui concerne les variables, chaque variable numérique occupe 7 octets, chaque chaîne occupe (7 + longueur) octets. Un tableau occupe x(n+1)+2d, où d est le nombre de dimensions, n est la taille du tableau (y compris l'élément N°Ø) et x=5 (nombres) ou 3 (chaînes de caractères).

On gagne de la place mémoire en supprimant tous les blancs inutiles, en mettant plusieurs instructions par ligne, en évitant les REM, en utilisant des variables plutôt que des constantes. Utilisez des GOSUB dès qu'il faut faire appel plusieurs fois à une séquence d'instructions identiques.

35 - Corriger un programme qui contient des mouvements de curseur

Tous les problèmes viennent de ce que, lorsque l'on veut un mouvement de curseur effectif, le système se trouve en mode guillemets et imprime donc un caractère mouvement de curseur différé (g, d etc...) ou vice-versa. Voici deux cas typiques :

- quand on vient de taper Inst, on est en mode guillemets dans la fenêtre d'insertion donc, si l'on essaie d'y mettre des caractères curseur, on obtiendra les caractères différés.

- quand on entre dans une zone entre guillemets, par mouvements de curseur au cours d'une correction, on n'est pas en mode guillemets. Ici par exemple, dans 1Ø PRINT "hXXX" on veut remplacer le h (listé ♥) par b (listé ●) il faut amener le curseur sur le ♥, faire Inst,b, dd pour amener le curseur sur le premier X, et enfin faire Del puis Return. Une autre façon aurait été d'amener le curseur sur le guillemet ouvrant et de retaper le guillemet.

Dans tous les cas où on est en mode guillemets alors qu'on ne le souhaitait pas, on s'en sort en faisant Return et en revenant sur l'instruction.

36 - Savoir ce qu'il y a sur une cassette

Il est conseillé de tenir un répertoire des cassettes à mesure qu'on les remplit. Mais si vous ne l'avez pas fait, il suffit de faire un VERIFY"BZZBZZ". BZZBZZ est supposé n'être le nom d'aucun des programmes de la cassette. A ce moment, le VIC va parcourir toute la cassette, et à chaque programme qu'il trouvera, il imprimera : FOUND nom. Et comme nom est différent de BZZBZZ, il continuera.

37 - Changer (par software) le numéro de périphérique IEEE d'une unité de disque

Si l'on veut que l'unité K (par exemple K=8) réponde au numéro de périphérique L (exemple : L=9), faire :

```
OPEN 15,K,15
```

puis

```
PRINT#15,"M-W"CHR$(X)CHR$(0)CHR$(2)CHR$(L+32)CHR$(L+64)
```

avec X = 119 sur 1540.

Attention : c'est un changement software et non hardware. Le numéro redevient K si l'on fait reset en éteignant et allumant l'unité de disques ou l'unité centrale, ou si l'on envoie la commande :

```
OPEN 1,L,15,"U:"
```

38 - Adapter les programmes aux différentes extensions mémoire

Selon la taille mémoire, un certain nombre d'adresses-mémoire changent. Ce sont :

	VIC de base	*ext.3K*	*ext.≥ 8K*
le début Basic	$1000 4096	$0400 1024	$1200 4608
Mémoire d'écran E	$1E00 7680	$1E00 7680	$1000 4096
Mémoire de couleur CO	$9600 38400	$9600 38400	$9400 37888

C'est pourquoi, dans tous les programmes de haute résolution, il faut paramétrer les variables E et CO.

D'autre part, avec une extension ≥ 8K, le mieux est de mettre le début de Basic à $2000 par

```
POKE 44,32:POKE 46,32:POKE 48,32:POKE 50,32:CLR
```

39 - Fixer les paramètres d'une communication RS232

Ouverture par OPEN 2,2,Ø,CHR$(V)+CHR$(W)

V= V1+32*V2+128*V3

V3	*bits d'arrêt*	
0	1	**
1	2	***

V2	*bits/mot*	
0	8	
1	7	***
2	6	
3	5	

V1	*vitesse*	
1	50	
2	75	
3	110	***
4	135	
5	150	
6	300	***
7	600	
8	1200	
9	1800	
10	2400	**

W = W1+16*W2+64*W3*W4+32*W3

W1	*interface*
0	3 lignes ***
1	X lignes

W2	*duplex*	
0	full	***
1	half	

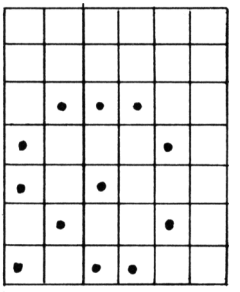
W3	*parité*
0	aucune
1	selon W4

W4	*parité*
0	impaire
1	paire
2	1 émis pas de contrôle en réception
3	0 émis pas de contrôle en réception

40 - Définir un caractère programmable sur VIC 1515

1 - On passe en mode graphique par PRINT#4,CHR$(8);

2- Ensuite chaque octet définit une colonne élémentaire du caractère ; par exemple pour un ç on enverrait :



1						
2						
4		•	•	•		
8	•				•	
16	•		•			
32		•			•	
64	•		•	•		
somme	88	36	84	68	40	0
128 +	216		212		168	
somme		164		196	128	

```
PRINT#4,CHR$(216)CHR$(164)CHR$(212)
        CHR$(196)CHR$(168)CHR$(128);
```

3 - On sort du mode graphique par

```
PRINT#4,CHR$(15)
```

(Les numéros sont les numéros des "trucs")

Adapter les programmes aux extensions mémoire..... 38
Adresse d'une variable 17
Affichage (augmenter largeur) 20
Arguments pour SYS 11
ASCII (correspond à la touche enfoncée) 30
Augmenter largeur affichage 20
Attendre un caractère 8
Attendre un STOP 24
Attendre Shift 7
Capacité (mesurer) 28
Caractère au clavier 24
Caractère programmable 21,40
Cassette (contenu) 36
Cassette (vérifier) 31
Changer numéro disque 37
Chargement sans translation 14
Clavier (attendre) 8,24
Clavier (déceler touche) 30
Clavier (vider tampon) 22
Connecter la télé 16
Contenu cassette 36
Corriger avec mouvements de curseur 35
Curseur (correction) 35
Curseur pendant GET 23
Curseur (positionner) 1
Décimales (imprimer) 12
Définir caractère programmable sur imprimante 40
Définir caractère programmable sur écran 21
Définir fonction en cours d'exécution 15
Disquette (changer numéro périphérique) 37
Ecran (effacer) 25

INDEX DES "COMMENT...?"

Ecran (scroll) 26
Effacer écran 25
Empêcher listing 3
Extension mémoire (s'adapter à) 38
Eviter ennuis de INPUT 33
Fixer les paramètres RS232 39
Fonction (définir en cours d'exécution) 15
Fonction (touches de) 5
GET avec curseur 23
GOTO calculé 11
Graphique (passer en mode) 19
Imprimante (lister sur) 18
Imprimer avec D décimales 12
Imprimer avec justification 13
Indenter 10
Inhiber touche Stop 6
INPUT (éviter ennuis) 33
INPUT (simuler) 2
Joysticks (lire) 27
Justifier 13
Largeur affichage (augmenter) 20
Lire les joysticks 27
LIST (sur imprimante) 18
Listing (pause) 9
Listing (empêcher) 3
Mémoire (s'adapter aux extensions) 38
Mémoire (protéger) 29
Mesurer une résistance en capacité 28
Minuscules (passer en mode) 19
Numéro de périphérique (changer) 37
Passer de graphique à minuscules 19
Pause dans un listing 9
Péritel 16

Positionner curseur 1
Prévoir la taille d'un programme 34
Protéger mémoire 29
Relancer programme 32
Résistance (mesurer) 28
RS232 (fixer les paramètres) 39
Sauver une table 4
Scroll écran 26
Shift (attendre) 7
Simuler INPUT 2
Sortir d'INPUT 33
Stop (inhiber) 6
SYS avec arguments 11
Table (sauver) 4
Taille (prévoir) 34
Tampon clavier (vider) 22
Télé (connecter la) 16
Touches de fonction 5
Touche-ASCII 30
Translation (chargement sans) 14
Vérifier fonctionnement cassette 31
Vider tampon-clavier 22

Achevé d'imprimer en juin 1983
sur les presses de l'imprimerie Laballery et C^ie^
58500 Clamecy
Dépôt légal : juin 1983

N° d'impression : 305029
N° d'édition : 86595-74-2
ISBN : 2-86595-074-3

mémento

Un livre destiné à se trouver en permanence à côté de votre VIC tandis que vous programmez ? Un livre qui vous fait accéder rapidement à toutes les informations de référence : syntaxe des commandes, codes caractères, messages d'erreurs, codes machine, brochages, bonnes adresses ? Un livre qui se termine par un recueil de 40 "trucs" utiles, les "Comment...?" ? Le livre qui vous évitera d'aller glaner de ci de là tous ces renseignements existe. C'est "Clefs pour le VIC".

CLEFS POUR LE VIC

Editions du P.S.I.
B.P. 86
77402 Lagny/Marne Cedex
France

ISBN : 2.86595.074.3

82 FF

ouverture Mariane de Nayer

Imprimé en France * *